
ADVANTEST®

株式会社アドバンテスト

R3261/3361

オプション81

コントローラ機能(レファレンス)

取扱説明書

MANUAL NUMBER OJA00 9406

当社の製品が外国為替および外国貿易管理法の規定により、戦略物資あるいは役務等に該当する場合、輸出する際には日本国政府の許可が必要です。

緒言

1. 取扱説明書の構成

構成	内容	備考
R3261/3361 オプション81 コントローラ機能 (ガイド) 取扱説明書	1. 使用開始前の準備 2. プログラムの作成 3. プログラムの実行 4. 応用操作	コントローラ機能を Q&A 形式で説明 します。
R3261/3361 オプション81 コントローラ機能 (レファレンス) 取扱説明書 (本書)	第1部 1. 概要 2.～5. 計測器に外部端末を接続した場合の 説明 6.～9. 計測器のみの場合の説明 第2部 1. オプション81 (BASIC GPIBコントローラ) 2. コマンドとステートメントの文法と解説 3. ビルトイン関数 APPENDIX A1.1 機能別コマンドとステートメント一覧 A1.2 機能別ビルトイン関数とグラフィック機能一覧 A1.3 パラメータ指定一覧 A1.4 エラーメッセージ一覧	コントローラ機能を 詳しく説明しま します。

(注1) コントローラ機能を初めて使用する方は、(ガイド)を読んでから(レファレンス)に進むと理解しやすくなります。

(注2) 本体(R3261/3361)の説明は、本体に付属の“R3261/3361シリーズ取扱説明書”を参照して下さい。

2. オプション81について

R3261/3361シリーズ用オプション81の内蔵コントローラ機能には、シリアルI/O(RS-232)のエディタ機能とパラレルI/Oを含んでいます。

3. 接続可能な外部端末について

R3261/3361シリーズ用オプション81に接続可能な外部端末は、VG-920およびVT-220相当品です。本書での説明には、VG-920を使用しています。

4. 本書中の略語について

略語	意味
PC	日本電気(株)製PC9801のパーソナル・コンピュータを表します。
HP	ヒューレット・パッカード社製のパーソナル・コンピュータを表します。
VG-920	ビクターデータシステムズ社製の端末を表します。
VT-220	DEC 製の端末を表します。
パソコン	パーソナル・コンピュータを表します。

第1部 a t eエディタ

第1部は、計測器に外部端末 (VG-920) を用いた場合と、計測器のみの場合に分けて a t eエディタを説明します。

- 計測器に外部端末 (VG-920) を用いた場合

- 2章 特徴
- 3章 起動
- 4章 機能一覧
- 5章 詳細解説

- 計測器のみの場合

- 6章 特徴
- 7章 起動
- 8章 機能一覧
- 9章 詳細解説

第 1 部の目次

1. 概要	1 - 1
2. 特徴 (計測器 + 外部端末の場合)	2 - 1
2.1 キーボード	2 - 1
2.2 ポップアップメニュー	2 - 1
2.3 BASIC実行環境	2 - 2
2.4 ラベルの定義	2 - 2
2.5 ファイル	2 - 3
2.6 ヘルプメニュー	2 - 3
3. 起動 (計測器 + 外部端末の場合)	3 - 1
4. 機能一覧 (計測器 + 外部端末の場合)	4 - 1
4.1 カーソル制御	4 - 1
4.2 挿入	4 - 1
4.3 削除	4 - 1
4.4 コピー、移動、削除 (範囲指定)	4 - 2
4.5 ウィンドウ	4 - 2
4.6 ファイル	4 - 2
4.7 検索	4 - 2
4.8 置換	4 - 2
4.9 BASICモード	4 - 2
4.10 ヘルプ	4 - 3
4.11 ポップアップメニュー	4 - 3
4.12 キャンセル	4 - 3
4.13 エディタの再起動	4 - 3
5. 機能解説 (計測器 + 外部端末の場合)	5 - 1
5.1 カーソル制御	5 - 1
5.2 挿入	5 - 2
5.3 削除	5 - 3
5.3.1 一文字削除	5 - 3
5.3.2 一行削除	5 - 4
5.3.3 範囲指定削除	5 - 5
5.4 コピー、移動、削除 (範囲指定)	5 - 6
5.4.1 テキストの削除	5 - 6
5.4.2 テキストの復旧、移動	5 - 7
5.4.3 テキストのコピー	5 - 7
5.5 ウィンドウ	5 - 9
5.5.1 ウィンドウの分割	5 - 9
5.5.2 ウィンドウの初期化	5 - 9
5.5.3 画面の再表示	5 - 10
5.6 ファイル	5 - 11
5.6.1 ファイルのセーブ	5 - 11
5.6.2 ファイルのロード	5 - 12
5.6.3 ファイルの更新	5 - 13
5.6.4 ファイルの消去	5 - 13
5.7 検索	5 - 14
5.8 置換	5 - 15

5.9	BASIC	5 - 17
5.9.1	行番号の設定	5 - 17
5.9.2	BASICの実行	5 - 19
5.9.3	BASICの停止	5 - 19
5.9.4	BASICモード	5 - 20
5.9.5	BASICの継続	5 - 20
5.10	ヘルプ	5 - 21
5.11	ポップアップメニュー	5 - 22
5.12	エディタ機能の中止	5 - 25
5.13	エディタの初期化	5 - 26
6.	特徴 (計測器のみの場合)	6 - 1
6.1	キーボード	6 - 1
6.2	ポップアップメニュー	6 - 2
6.3	BASIC実行環境	6 - 3
6.4	ラベルの定義	6 - 3
6.5	ファイル	6 - 4
6.6	文字入力	6 - 4
6.7	ヘルプメニュー	6 - 4
7.	起動 (計測器のみの場合)	7 - 1
8.	機能一覧 (計測器のみの場合)	8 - 1
8.1	カーソル制御	8 - 1
8.2	挿入	8 - 1
8.3	削除	8 - 1
8.4	コピー、移動、削除 (範囲指定)	8 - 2
8.5	ウィンドウ	8 - 2
8.6	ファイル	8 - 2
8.7	検索	8 - 2
8.8	置換	8 - 2
8.9	BASICモード	8 - 3
8.10	ポップアップメニュー	8 - 3
8.11	キャンセル	8 - 3
8.12	エディタの再起動	8 - 3
9.	機能解説 (計測器のみの場合)	9 - 1
9.1	カーソル制御	9 - 1
9.2	挿入	9 - 2
9.3	削除	9 - 3
9.3.1	一文字削除	9 - 3
9.4	コピー、移動、削除 (範囲指定)	9 - 4
9.4.1	テキストの削除	9 - 4
9.4.2	テキストの復旧、移動	9 - 5
9.4.3	テキストのコピー	9 - 5
9.5	ウィンドウ	9 - 7
9.5.1	ウィンドウの分割	9 - 7
9.5.2	ウィンドウの初期化	9 - 7
9.5.3	画面の再表示	9 - 8

9.6	ファイル	9 - 9
9.6.1	ファイルのセーブ	9 - 9
9.6.2	ファイルのロード	9 - 10
9.6.3	ファイルの更新	9 - 11
9.6.4	ファイルの消去	9 - 12
9.6.5	プログラム・オート・スタート機能	9 - 12
9.7	検索	9 - 14
9.8	置換	9 - 15
9.9	BASIC	9 - 17
9.9.1	行番号の設定	9 - 17
9.9.2	BASICの実行	9 - 19
9.9.3	BASICの停止	9 - 19
9.9.4	BASICモード	9 - 20
9.9.5	BASICの継続	9 - 20
9.10	ヘルプ	9 - 21
9.11	ポップアップメニュー	9 - 22
9.12	エディタ機能の中止	9 - 24
9.13	エディタの初期化	9 - 25

1. 概要

ateエディタは、計測器上でBASICプログラミングを容易に行なえるように開発されたフルスクリーンエディタです。

計測器の背面パネルにあるRS-232 コネクタに接続した外部端末(VG-920)を用いて編集でき、操作性が大変向上しました。また、計測器のみでも編集でき、現場でのプログラムの修正などポータブル性も兼ね備えています。

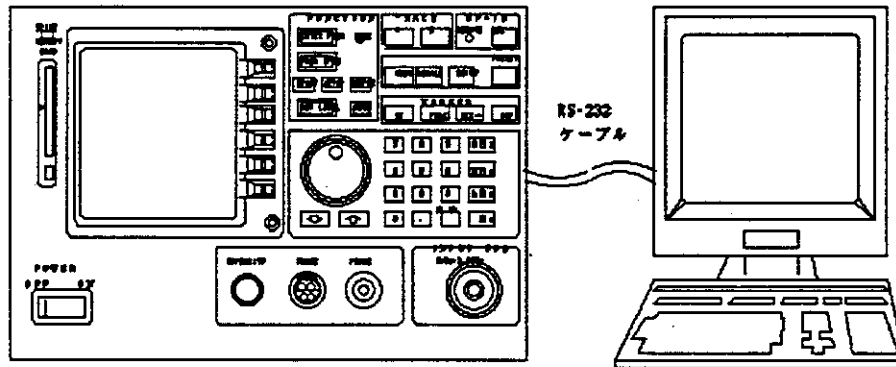


図1-1 R3261/3361とVG-920の接続

MEMO 

2. 特長（計測器+外部端末の場合）

ateエディタは、高度な編集機能を備えたフルスクリーンエディタです。機能としてカーソル制御、テキストの挿入、削除、コピー、移動、置換、検索、ウィンドウ、ファイル、BASICプログラムの実行などがあります。また、これらの機能を円滑にするためにポップアップメニュー、ヘルプメニューがあります。

2. 1 キーボード

ateエディタの基本的な機能は端末のキーボードに割り当てられています。複雑な編集作業が簡単なキー操作でできます。

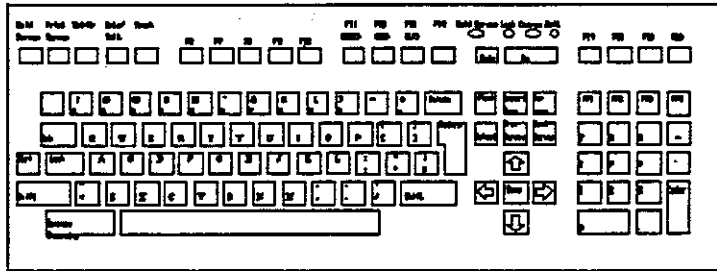


図2-1 VG-920のキーボード

2. 2 ポップアップメニュー

ateエディタの機能すべてがポップアップメニューで実行できます（カーソル移動は除く）。実行したい機能をポップアップメニュー内よりカーソルキーで選択し、<Enter>キーを押すと、その機能が実行されます。

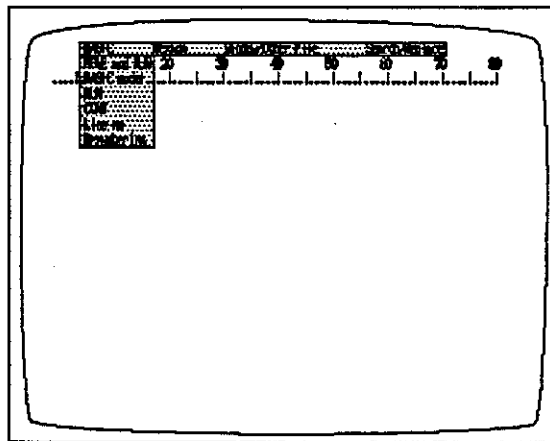


図2-2 ポップアップメニュー

2. 3 BASIC実行環境

ateエディタで編集したBASICプログラムをそのままエディタ上で実行できます。BASICのRUN, CONT, SCRATCHコマンドは、キーボードまたはポップアップメニューによって実行します。それ以外のコマンドはBASICモードのミニウィンドウ内で実行します。

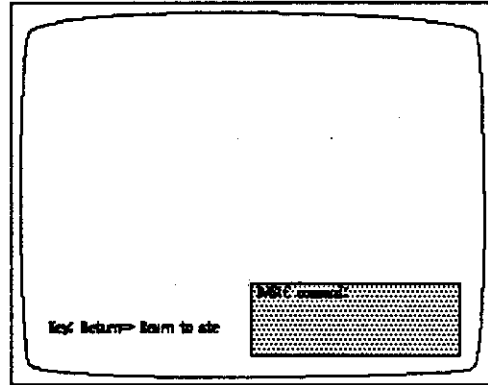


図2-3 BASICモード

2. 4 ラベルの定義

ateエディタでは、行番号は省略し、ラベルを定義します。これにより、サブルーチン・コールなどを使うとき、わかりやすいプログラムを作成できます。

注) ラベルは文字列の先頭に必ず* (アスタリスク) をつけます。

```
.....|.....1.....|.....2
A=0
*LABEL
A=A+1
IF A >= 10 THEN
    GOTO *ABCD
END IF
GOTO *LABEL
*ABCD
```

図2-4 ラベルの定義

2. 5 ファイル

作成したプログラムは、ファイル単位で本体のメモリ・カードへセーブできます。また、メモリ・カードからファイルをロードして、編集もできます。

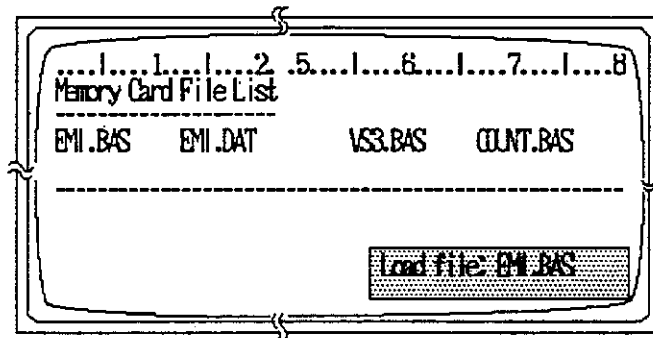


図2-5 ファイルのロード

2. 6 ヘルプメニュー

VG-920のキーボードに割り当てられている機能が一覧表示されます。

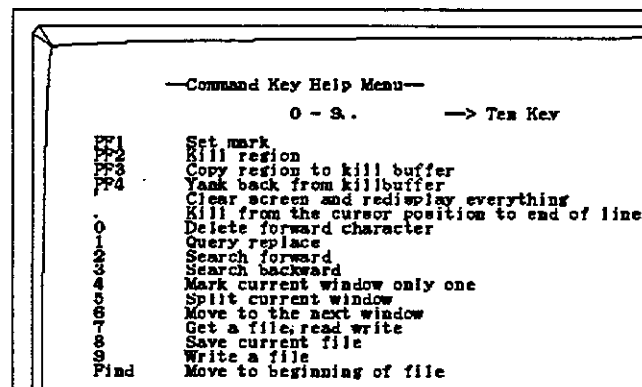


図2-6 ヘルプメニュー

MEMO 

3. 起動 (計測器+外部端末の場合)

(1) 操作

計測器に端末 (VG-920) を接続し、電源を入れると、以下に示す初期画面が表示され、ate エディタが起動します。

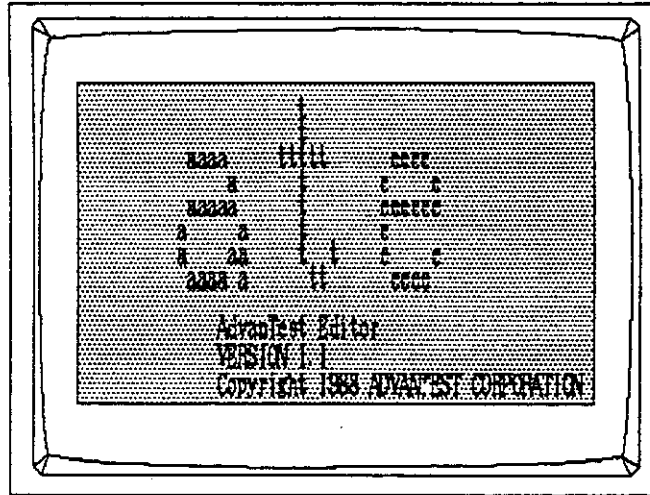


図3-1 ateの初期画面

次に何かキーを押すと、以下に示す画面に変わります。(メッセージ・ライン、ミニウインドウは、表示されません。)

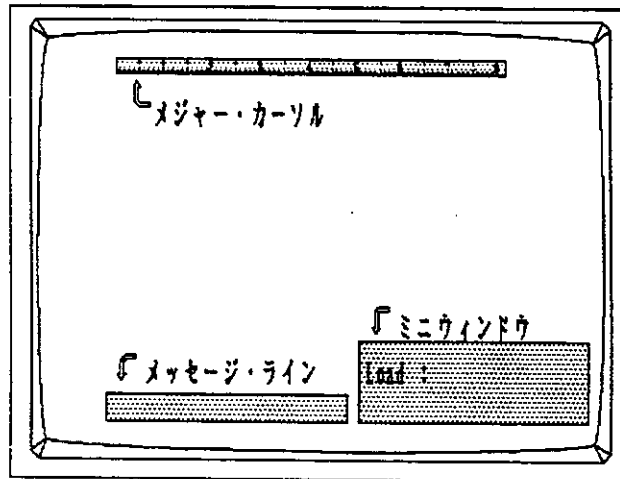


図3-2 ateの動作画面

(2) 画面表示の説明

画面の表示機能を説明します。(図3-2 ateの動作画面 参照)

- ・メジャーカーソル
編集時にカーソル位置を把握するためのメジャー表示です。
- ・メッセージライン
コマンドの実行手順、実行結果、エラーメッセージを表示します。
- ・ミニウインドウ
コマンドを実行する場合、パラメータが必要なときに使います。バッファ名、ファイル名、文字列などパラメータを要求するコマンドに応じて様々です。
表示されるプロンプトに従ってパラメータを入力します。パラメータの後に必ず<Return>キーを入力して下さい。プロンプトの後ろの[]で囲まれた部分はデフォルトパラメータです。このデフォルトはパラメータを入力しないで<Return>キーを押すとパラメータになります。

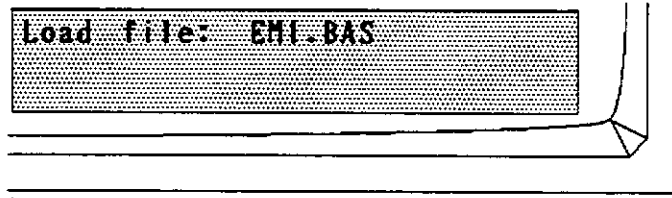


図3-3 ミニウインドウ

4. 機能一覧 (計測器+外部端末の場合)

ateエディタの各機能は、一部を除いてキーボード上にメイン・キーボード、編集キーボード、補助キーボード、ファンクション・キーボードと割り当てられています。

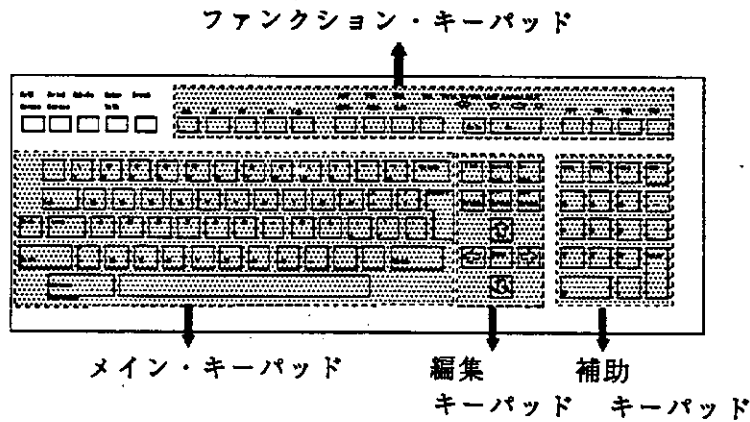


図4-1 キーボード配置図

4.1 カーソル制御

→	次の文字に移動	(編集キーボード)
←	前の文字に移動	(編集キーボード)
↑	前の行に移動	(編集キーボード)
↓	次の行に移動	(編集キーボード)
Find	ファイルの先頭に移動	(編集キーボード)
Insert-Here	ファイルの末尾に移動	(編集キーボード)
Remove	行の先頭に移動	(編集キーボード)
Select	行の末尾に移動	(編集キーボード)
Prev-screen	前に一画面分移動	(編集キーボード)
Next-screen	次に一画面分移動	(編集キーボード)

4.2 挿入

Return	改行文字を挿入	(メイン・キーボード)
Tab	タブを挿入	(メイン・キーボード)
上記以外のメインキーボード	一般文字	

4.3 削除

Delete	カーソル前の文字を削除	
0	カーソル位置の文字を削除	(補助キーボード)
	カーソル位置から行末までを削除	(補助キーボード)
	で削除した文字はPF4キー (補助キーボード) を押すと元に戻せます。	

4. 4 コピー、移動、削除（範囲指定）

PF1	リージョン処理に必要なマークをカーソル位置にセットする （補助キーパッド）
PF2	マークからカーソル位置までを削除し、バッファに入れる （補助キーパッド）
PF3	マークからカーソル位置までをバッファに入れる （補助キーパッド）
PF4	PF2キーまたはPF3キーによりバッファに入れられた部分をカーソル位置にコピーする（補助キーパッド）

4. 5 ウィンドウ

4	ウィンドウを1つにする（補助キーパッド）
5	ウィンドウを2つに分割する（補助キーパッド）
6	カーソルを次のウィンドウに移動（補助キーパッド）
.	画面をクリアして再表示する（補助キーパッド）

4. 6 ファイル

7	メモ리카ードから指定したファイルをロードする （補助キーパッド）
8	メモ리카ードにファイルをセーブする （補助キーパッド）
9	メモ리카ードに指定したファイル名でセーブする （補助キーパッド）
-	ファイル名一覧ウィンドウへカーソルを移動する （補助キーパッド）

4. 7 検索

2	前方に文字列を検索（補助キーパッド）
3	後方に文字列を検索（補助キーパッド）

4. 8 置換

1	文字列を置き換え（補助キーパッド）
---	-------------------

4. 9 BASICモード

F13	自動的な行番号挿入を設定（ファンクションキーパッド）
F14	行番号の再定義（ファンクションキーパッド）
F17	プログラムをBASICバッファに転送後、実行する （ファンクションキーパッド）
F18	BASICモードへ移行する（ファンクションキーパッド）
F19	すでに転送済みのプログラムを実行する （ファンクションキーパッド）
F20	Ctrl-Cで中断したプログラムを続行する （ファンクションキーパッド）

4. 1 0 ヘルプ

Help エディタ機能とキー割当の一覧 (ファンクションキーパッド)

4. 1 1 ポップアップメニュー

Do (Ctrl-A) ポップアップメニューを表示する (ファンクションキーパッド)

4. 1 2 キャンセル

F11 (Ctrl-Z) エディタの機能を中止する (ファンクションキーパッド)

4. 1 3 エディタの再起動

F12 現エディタを終了し、バッファを初期化して再起動する
(ファンクションキーパッド)

MEMO 

5. 機能解説（計測器＋外部端末の場合）

5. 1 カーソル制御

カーソルは編集位置を示すものであり、編集はカーソルを基準にして行なわれます。

カーソル制御とは、画面上でカーソルを特定の場所に移動することを言います。まず、1文字ずつ上下左右に移動する機能があります。これらの機能はそれぞれカーソルキーに割り当てられており、現在の場所から新しい場所へカーソルを移動します。

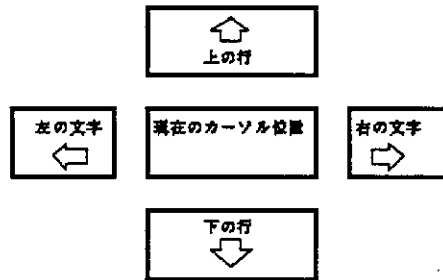


図5-1 カーソルキー

これらは基本的なカーソル移動機能を持ち、頻繁に使います。画面の上限や下限を超えて移動しようとする時、その方向にあるテキストがスクロールし、カーソルは常に画面内に存在します。

この他に編集キーボードに割り当てられている機能があります。

Find	ファイルの先頭に移動する
Insert-Here	ファイルの末尾に移動する
Remove	行の先頭に移動する
Select	行の末尾に移動する
Next-Screen	次のページに進む
Prev-Screen	前のページに戻る

図5-2 カーソル移動に関する補助キーボード

5. 2 挿入

ateエディタでは、常にテキストの挿入ができる状態になっているので、キーボードを叩くと制御文字以外はすべて挿入されます。文字は一般文字と制御文字の2つに分けられます。一般文字とは'A', '1'など目に見える文字を言い、制御文字とは'ESC', 'CTRL'など目に見えない文字を言います。

一行が画面の幅以上に長くなると、右端に'\$'記号を表示して、その行が継続されていることを示します。

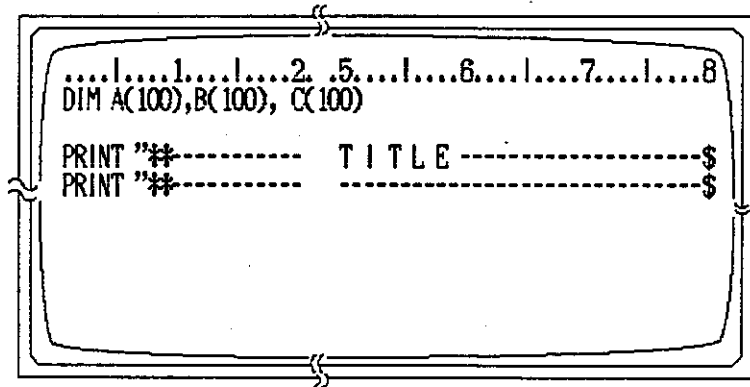


図5-3 文字の継続を表す

行の終わりは<Return>キーを入力すると、改行文字で表わされます。また、文字の位置を調節するタブ機能は<Tab>キーを入力します。タブサイズは8文字分です。

Return	改行文字を挿入する
Tab	タブを挿入する
上記以外のメインキーパッド	一般文字を挿入する

図5-4 挿入に関するメイン・キーパッド

5.3 削除

削除には、一文字削除、一行削除、範囲指定削除があります。これらの機能を説明します。

5.3.1 一文字削除

一文字削除には、カーソル直前の文字を削除するものと、カーソル位置の文字を削除するものの2通りの方法があります。

(1) カーソル直前の文字を削除する場合

カーソル直前の文字を削除するには、<Delete>キー（メイン・キーボード）を入力します。

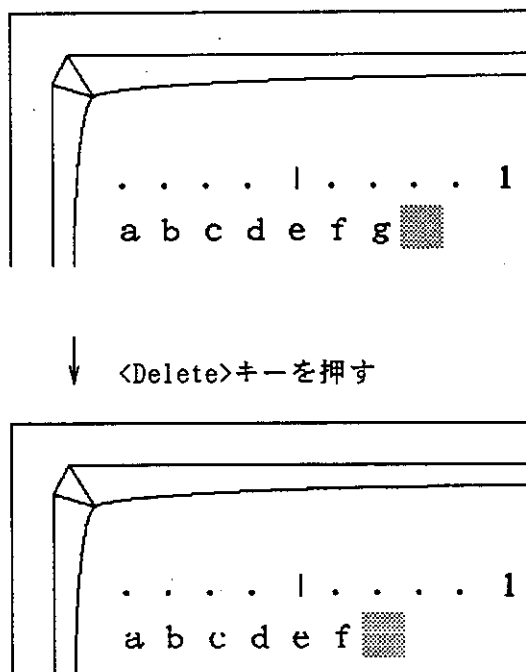


図5-5 カーソル直前の文字を削除

- (2) カーソル位置の文字を削除する場合
カーソル位置の文字を削除するには、<0>キー（補助キーパッド）を入力します。

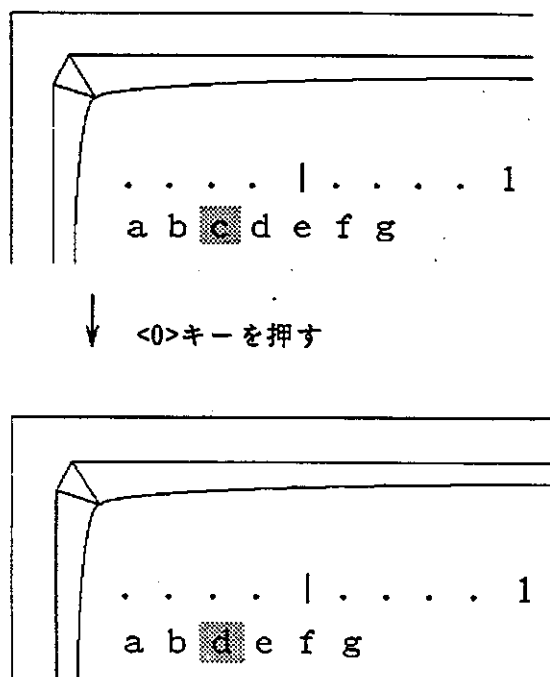


図5-6 カーソル位置の文字を削除

5.3.2 一行削除

カーソル位置から行末までを削除するには、<>キー（補助キーボード）を入力します。最初の<>キーでその行の内容が削除され、再度<>キーを入力すると、その行自身が削除されます。

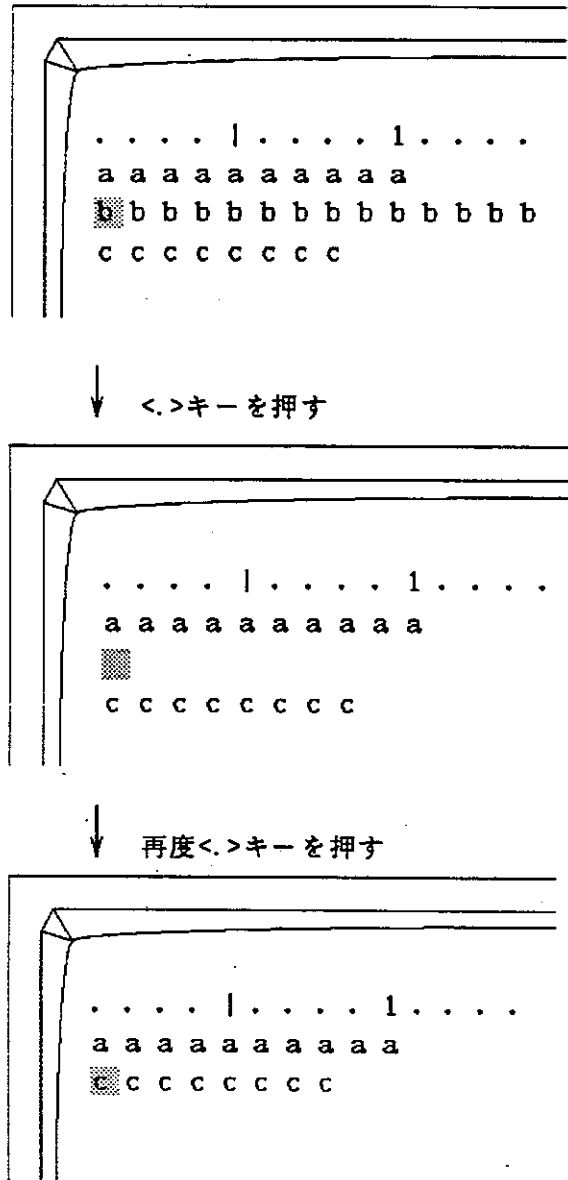


図5-7 カーソルから行末までを削除

5.3.3 範囲指定削除

範囲を指定して削除する方法は、テキストの保存と関連があるため[5.4節]で説明します。

Delete	カーソル直前の文字を削除
0	カーソル位置の文字を削除（補助キーボード）
.	カーソルから行末までを削除（補助キーボード）

図5-8 削除に関するキーボード

5. 4 コピー、移動、削除（範囲指定）

コピー、移動、削除をするために範囲を指定して、テキストを保存します。

5. 4. 1 テキストの削除

(1) マークセット

削除、移動をするときは、削除する範囲の先頭位置にカーソルを移動して、**<F2>**キー（補助キーパッド）またはポップアップメニューの**"Mark set"**でマークを設定します。

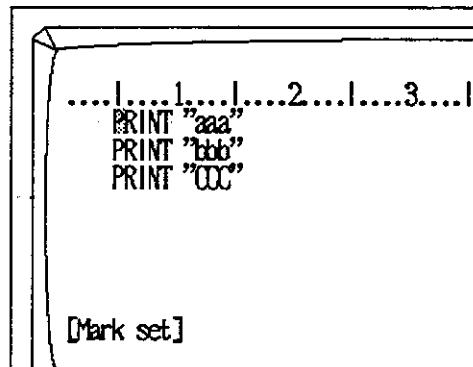


図5-9 マークセット

(2) テキストの削除

削除する範囲の最終位置よりも1文字分先へカーソルを移動し、**<F2>**キー（補助キーパッド）またはポップアップメニューの**"Delete region"**を入力します。すると、(1)でマークをセットした位置から現在のカーソル位置の手前までのテキストを削除します。

削除したテキストは、内部バッファに保存されているので、後述のテキストの移動、復旧、コピーなどは(1)、(2)の操作をした後、実行します。

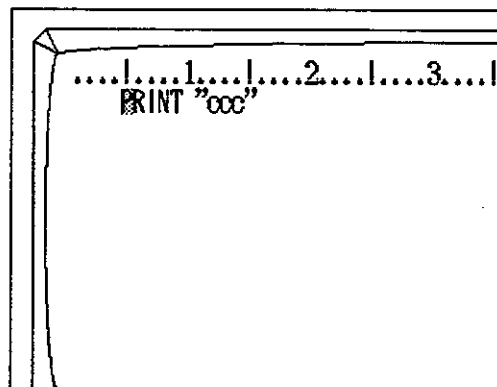


図5-10 テキストの保存（削除）

5. 4. 2 テキストの復旧、移動

削除したテキストを元に戻すには、現在のカーソル位置で<Ctrl>キー（補助キーパッド）またはポップアップメニューの"Ctrl"を入力します。テキストを移動する場合は、移動したい位置へカーソルを移動し、<Ctrl>キー（補助キーパッド）またはポップアップメニューの"Ctrl"を入力します。

このように<Ctrl>キー（補助キーパッド）と、ポップアップメニューの"Ctrl"は、保存したテキストを現在のカーソル位置に挿入する機能です。

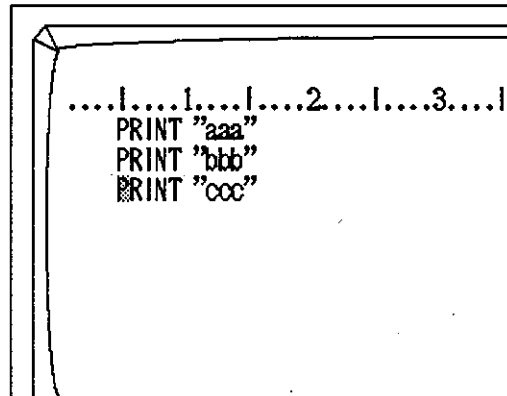


図5-11 テキストの復旧

5. 4. 3 テキストのコピー

テキストをコピーするには、範囲の先頭位置を削除する場合と同様に設定し、最終位置にカーソルを移動し、<Ctrl>キー（補助キーパッド）またはポップアップメニューの"Copy"を入力します。すると範囲内は何も変わったように見えませんが、指定したテキストが内部バッファにセーブされているので、コピー位置にカーソルを移動し、<Ctrl>キー（補助キーパッド）またはポップアップメニューの"Ctrl"を入力します。

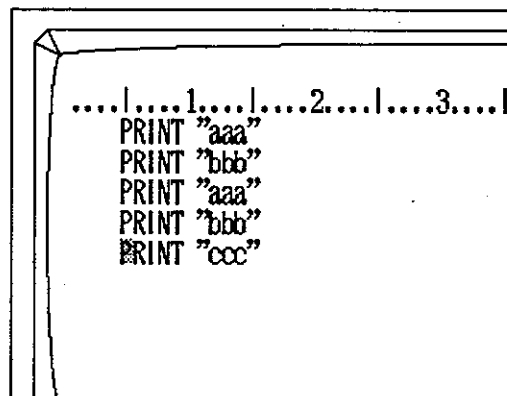


図5-12 テキストのコピー

<Ctrl>キー（補助キーパッド）またはポップアップメニューの"Ctrl"によって復旧されるテキストは、内部バッファ内の最新のテキストになります。

PF1	範囲指定に必要なマークをカーソル位置にセットする
PF2	マークからカーソル位置までを削除し、テキストを保存する
PF3	マークからカーソル位置までのテキストを保存する
PF4	保存したテキストをカーソル位置に挿入する

図5-13 リージョン処理に関する補助キーパッド

5. 5 ウィンドウ

通常のウィンドウはオーバーラッピング型とタイル型に分けられますが、ateエディタのウィンドウは画面を上下に分割するタイル型になります。

通常のウィンドウは1つですが、ateエディタは2つのウィンドウを持つことができ、それぞれに対してテキストを表示します。例えば、画面内に表示しきれない2つの部分を同時に参照できます。

注) オープンできるファイルは1つに限られているため、2つのウィンドウにそれぞれ別のファイルをオープンできません。

5. 5. 1 ウィンドウの分割

ウィンドウを2つに分割するには、`<F10>`キー(補助キーパッド) またはポップアップメニューの"`分割`"を入力します。

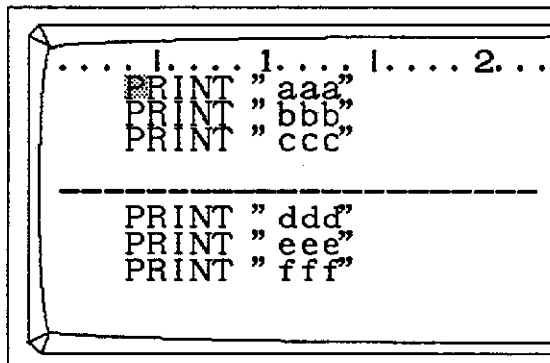


図5-14 上下に分割されたウィンドウ

5. 5. 2 ウィンドウの初期化

上下に分割されたウィンドウを1つに戻すには、`<F10>`キー(補助キーパッド) またはポップアップメニューの"`閉じる`"を入力して、残したいウィンドウにカーソルを移動します。次に、`<F10>`キー(補助キーパッド) またはポップアップメニューの"`閉じる`"を入力すると、ウィンドウは1つに戻ります。

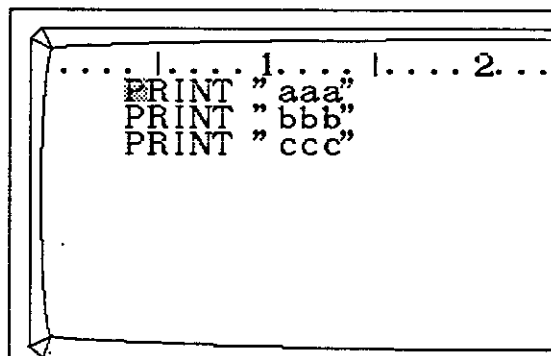


図5-15 ウィンドウを1つにする

5.5.3 画面の再表示

編集中のテキストを再表示したいときは、<>キー（補助キーボード）またはポップアップメニューの"Redisplay"を入力して、テキストを再表示します。

4	他のウィンドウを消去して、カーソルのあるウィンドウ 1つにする
5	ウィンドウを上下に分割し、カーソルを上ウィンドウ に置く
6	カーソルを次のウィンドウに移動する
,	画面をクリアして、テキストの再表示を行なう

図5-16 ウィンドウに関する補助キーボード


5. 6 ファイル

ateエディタでは、テキストをファイルという単位として、本体のメモリ・カードへのセーブ、メモリ・カードからのロードができます。編集したテキストは必ずメモリ・カードへセーブするようにして下さい。

注) セーブしないままエディタを終了すると、編集したテキストは失われます。

ここでは、ファイルのセーブ/ロードを説明します。

5. 6. 1 ファイルのセーブ

編集したテキストをメモリ・カードにセーブするには、<F5>キー（補助キーパッド）またはポップアップメニューのを入力します。ここでセーブするファイルに名前をつけます。ファイル名は 最大10文字です。

セーブが終了すると、書き込んだ行数をメッセージラインに表示します。

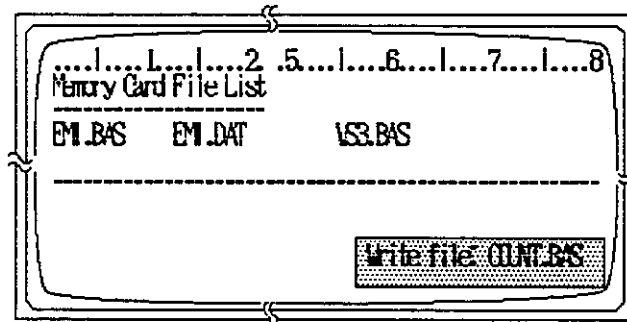


図5-17. テキストのセーブ（ファイル名を入力）

5. 6. 2 ファイルのロード

セーブされているファイルをロードするには、**<F2>**キー（補助キーボード）またはポップアップメニューの**<Load>**を入力します。すると、画面上に現在メモリ・カードにセーブされているファイルが一覧表示され、どのファイルをロードするかを尋ねてきます。ここでファイル名を入力するか、または一覧表示されているウィンドウへ**<F2>**キー（補助キーボード）を押してカーソルを移動させます。次に、カーソルキーにてファイル名を選択し、最後に**<Return>**キーを入力します。

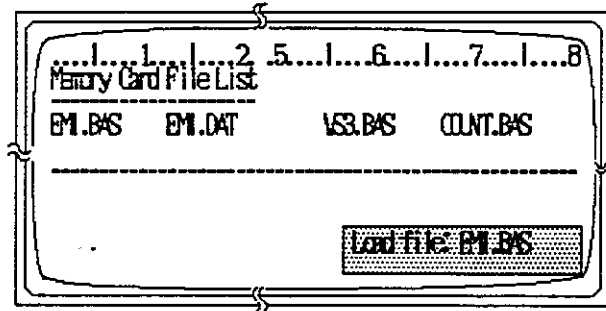


図5-18 ファイル名の入力

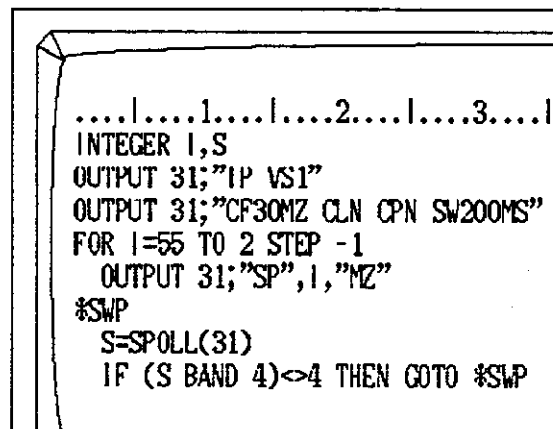


図5-19 ファイルのロード

メモリ・カードからファイルをロードしている最中は、本体のメモリ・カード挿入口上部のLEDが点灯します。ロードが終了すると、画面上にテキストが表示され、メッセージラインに読み込んだ行数が表示されます。

この時、行番号がついているものは自動行番号挿入(AUTO)機能が設定されます。

また、新しいファイルを作成するときは、そのファイル名を入力します。

メモリ・カードが挿入されていないときは、ブザー音またはエラーメッセージが表示されます。

5.6.3 ファイルの更新

同じファイル名でセーブするには、<8>キー(補助キーパッド)またはポップアップメニューの"Save"を入力します。すると、セーブするかどうかを尋ねてきます。テキストに何の変更もない場合は"No change"というメッセージを表示します。

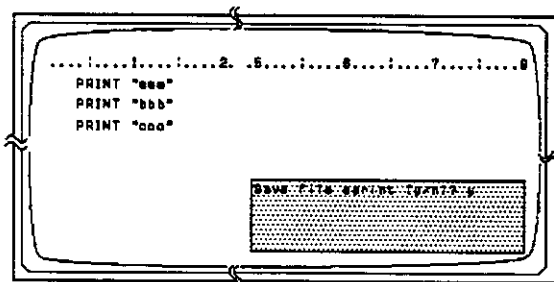


図5-20 テキストの更新(同一ファイル名)

<y> <Return>と入力すると、テキストはファイルに書き出されます。セーブが終了するとそのファイルの行数がメッセージラインに表示されます。

<n> <Return>と入力すると、テキストのセーブは実行しません。

7	メモリ・カードから指定したファイルをロードする
8	メモリ・カードに同一ファイル名でセーブする
9	メモリ・カードに指定したファイル名でセーブする
-	ファイル表示一覧のウィンドウへ移動する

図5-21 ファイルに関する補助キーパッド

5.6.4 ファイルの消去

第2部システム・コントローラの2章コマンドとステートメントの文法と解説 (58) PURGEを参照して下さい。

5. 8 置換

カーソル位置以降のテキスト中の任意の文字列を変更する機能です。

それでは、文字列を置き換えてみます。<F10>キー（補助キーパッド）またはポップアップメニューの“Query replace”を入力します。画面にミニウィンドウが表示され、置換したい文字列を尋ねてきたら、対象となる文字列を入力します。

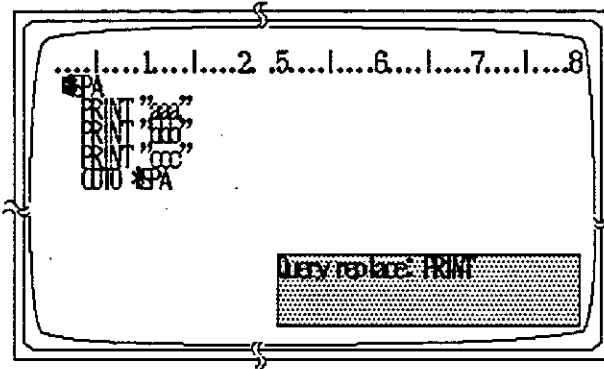


図5-25 置換対象文字列の入力

次に新たに変更する文字列を尋ねてきたら、任意の文字列を入力します。

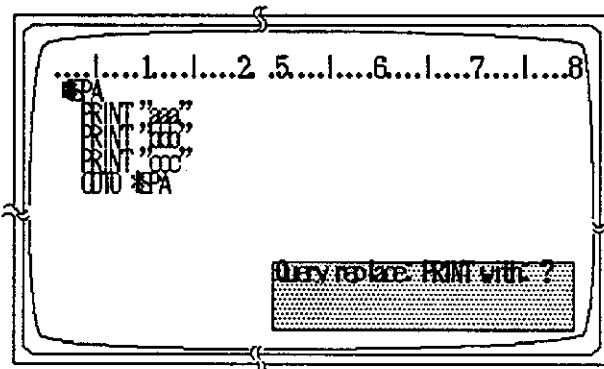


図5-26 置換文字列の入力

該当する文字列を検索すると、カーソルがその文字列の先頭に移動し、置換するかどうか待ち状態になります。ここで置換するときには<space>キー（メイン・キーパッド）を入力し、置換しないときには<Esc>キー（メイン・キーパッド）を入力します。また、置換して終了するときには<F10>キー（補助キーパッド）を入力し、置換しないで終了するときには<F11>（ファンクション・キーパッド）を入力します。

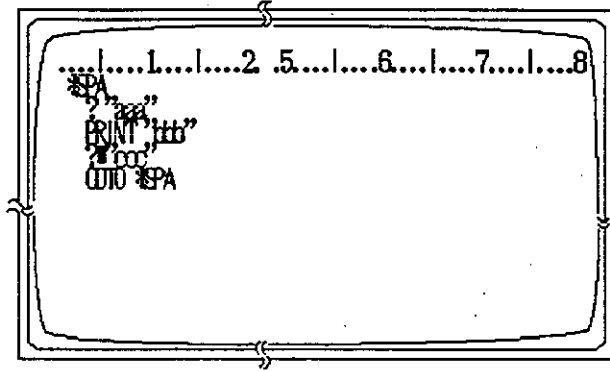


図5-27 実行結果

1	文字列を置き換える
space	置換する
delete	置換しない
.	置換して、終了する
F11	終了する

図5-28 置換に関する補助&ファンクション・キーパッド

5. 9 BASIC

ateエディタは、計測器上でのBASICプログラミングを行なうために開発されたポータブル・エディタです。よって編集しながらBASICの実行、デバッグが容易にできます。その他にBASICモードに移行してから様々なBASICコマンドの実行ができます。

ここではエディタ上から実行できるBASICコマンドを説明します。

5. 9. 1 行番号の設定

今までBASICプログラムを作成するときは行番号を使いました。ateエディタでは行番号の代わりにラベルを用いて、プログラムを分かりやすくします。

```
A=1:B=0
*LOOP
    B=B+A^2
    IF B>10000000 THEN GOTO *ENDLOOP
    PRINT B
    GOTO *LOOP
*ENDLOOP
STOP
```

図5-29 ラベルを用いたプログラム

また、自動行番号挿入 (AUTO) 機能により行番号をつけながら編集もできます。<F10>キー (ファンクション・キーパッド) またはポップアップメニューの "Line no." を入力すると、ミニウィンドウが表示され、開始番号と間隔を指定します。これで編集時に <Return> キーを入力すると、行番号が自動的に出力されます。

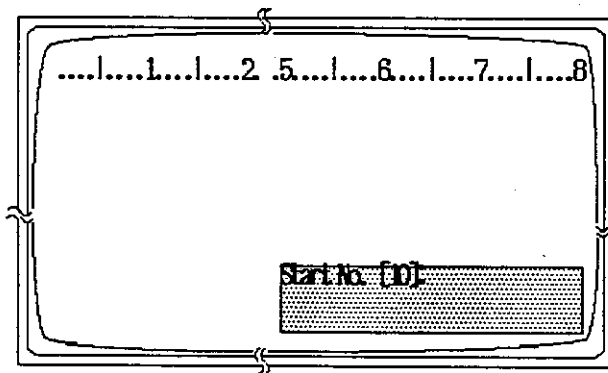


図5-30 開始番号の指定

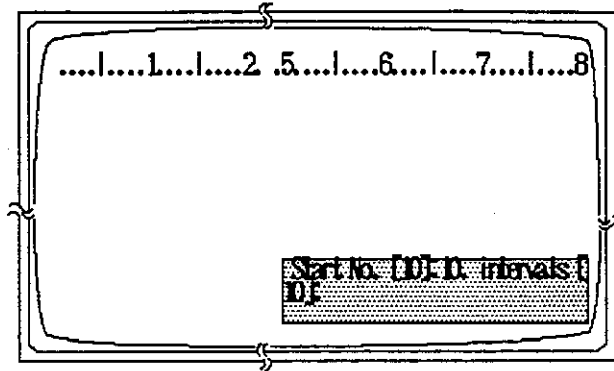


図5-31 行番号間隔の指定

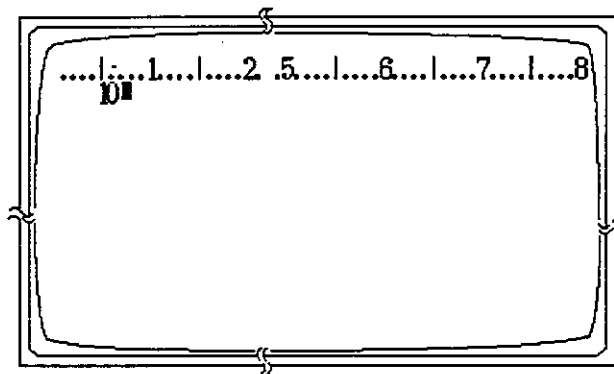


図5-32 自動行番号機能

また、編集途中の場合でも、この機能により行番号を追加することができます。行間での挿入を行なったときは、上の行番号にプラス1した数が行番号となります。

注) 一度この機能を実行すると、エディタの初期化(5.13節)をしない限り継続されます。

自動行番号挿入機能と同様、行番号のリナンバリング機能があります。これは行番号を一定の指定にしたがって付け直すものです。<F10>キー(ファンクション・キーパッド)またはポップアップメニューの"Renumbering"を入力すると、自動行番号挿入機能と同様に、開始番号と間隔を指定されます。

5. 9. 2 BASICの実行

プログラムを実行する前に、ateエディタとBASICインタプリタの関係を説明します。
プログラムの編集はateエディタで制御し、完成したプログラムはateエディタの内部バッファに格納されています。これとは別にBASICインタプリタも内部バッファを持っていて、実行するためにはこのバッファにBASICプログラムが存在しなければなりません。

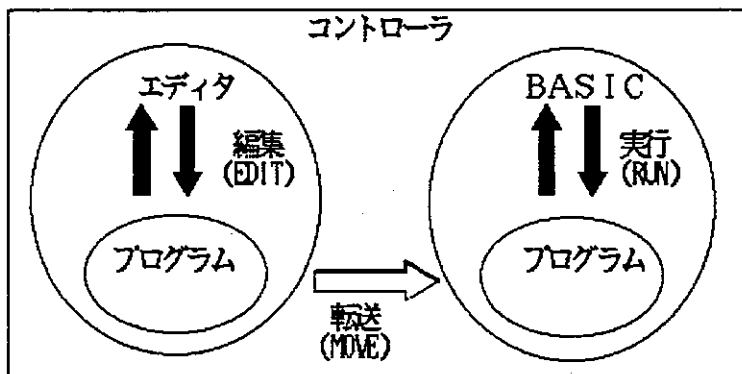


図5-33 エディタとインタプリタとの関係

よってBASICプログラムを実行する前には、エディタからインタプリタにプログラムを転送します。〈F10〉キー（ファンクション・キーパッド）またはポップアップメニューの"MOVE and RUN"を入力すると、エディタからプログラムが転送され、実行します。この時、前回実行したプログラムは消去されます。また、実行中にエラーが起こると、エディタに戻り、カーソルがエラー行に移動します。

すでに転送済みのプログラムを実行するときは、〈F10〉キー（ファンクション・キーパッド）またはポップアップメニューの"RUN"を入力します。プログラムを転送する前に〈F10〉キー（ファンクション・キーパッド）またはポップアップメニューの"MOVE and RUN"を実行したときは、"Program is not exist"というエラーメッセージが表示されます。

実行が終了すると、ミニウィンドウに"BASIC command:"のプロンプトが表示され、デバッグ環境になります。このままエディタへ戻るには、〈F10〉を入力します。

5. 9. 3 BASICの停止

実行中のBASICを停止するには、〈Ctrl-C〉（メイン・キーパッド）を入力します。

5. 9. 4 BASICモード

BASICモードとは、BASICに対してのコマンドを実行できる環境を言います。たとえば、前述のデバッグ環境です。〈F13〉キー（ファンクション・キーボード）またはポップアップメニューの“BASIC mode”を入力すると、ミニウィンドウが表示されます。

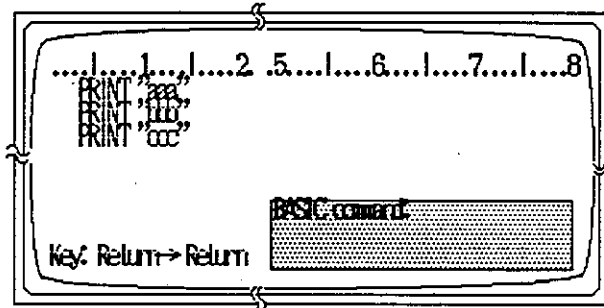


図5-34 BASICモード

実行できるコマンドは、RUN, LIST, CONTなどです。（コマンドの説明は [第2部システム・コントローラ] を参照）

注) プログラムの内容を変更する場合は必ずエディタに戻ってから行ないます。

5. 9. 5 BASICの継続

〈F11〉（メイン・キーボード）またはPAUSE命令で一時停止したプログラムを続けて実行するときは、〈F20〉キー（ファンクション・キーボード）またはポップアップメニューの“CONT”を入力します。これはBASICコマンドの“CONT”と同じです。停止した次の文から実行し、実行するプログラムがない場合は、“Program cannot be continued”というエラーメッセージが表示されます。

F13	自動的に行番号を挿入する
F14	行番号のリナンバリング
F17	プログラムを転送後、実行する
F18	BASICモードへ移行する
F19	すでに転送済みのプログラムを実行する
F20	Ctrl-CまたはPAUSE命令で中断したプログラムを 続行する

図5-35 BASICに関するファンクション・キーボード

5.10 ヘルプ

エディタの機能とキーボード (VG-920) の関係が画面上で確認できます。

<HELP>キー (ファンクション・キーパッド) またはポップアップメニューの"Help"を入力すると、どのキーがどんな機能を実行するか、簡単な説明が表示されます。

メニュー画面のスクロールなどカーソルを移動させる機能は編集機能と同様です。ただし左右の移動機能はありません。

<F11> (ファンクション・キーパッド) または<Ctrl-Z>キー (メイン・キーパッド) を入力すると、ヘルプ機能を中止し、元の編集画面に戻ります。

```

--Command Key Help Menu--
      D - 0..      --> Top Key

FF1      Set mark
FF2      Kill region
FF3      Copy region to kill buffer
FF4      Yank back from killbuffer
.        Clear screen and redisplay everything
.        Kill from the cursor position to end of line
0        Delete forward character
1        Query replace
2        Search forward
3        Search backward
4        Mark current window only one
5        Split current window
6        Move to the next window
7        Get a file, read write
8        Save current file
9        Write a file
Find     Move to beginning of file
```

図5-36 ヘルプメニュー

HELP	ヘルプメニューを表示する
F11 (Ctrl-Z)	ヘルプメニューを中止する

図5-37 ヘルプに関するファンクション&メイン・キーパッド

5. 1 1 ポップアップメニュー

前項まで説明してきたateエディタの機能は、すべてキーボードを操作して実行しました。これから説明するポップアップメニューを用いても同様の操作ができます。

カーソルキー以外のエディタ機能はすべてポップアップメニュー内に表示され、その中から選択し、実行します。これによりキー割当てがわからないときや、本体単独で使うときなどにとても役立ちます。

<F10> (ファンクション・キーパッド) または <Ctrl-F10> キー (メイン・キーパッド) を入力すると、画面最上部にポップアップメニューが表示されます。

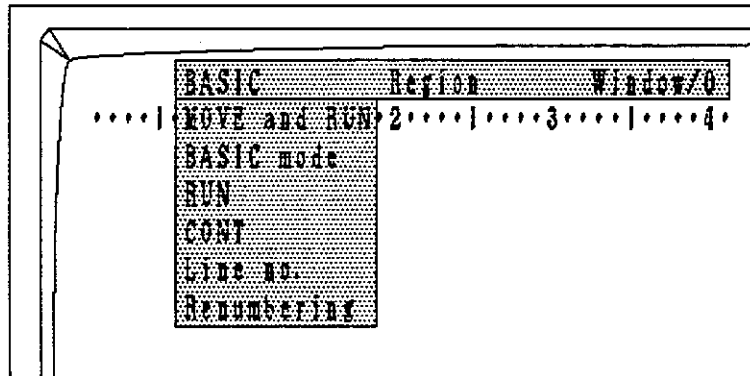


図5-38 ポップアップメニュー

BASIC	Region	Window/Other	File	Search/Replace
MOVE and RUN	Set mark	Only	Load	Forward search
BASIC mode	Kill region	Split	Save	Backward search
RUN	Copy region	Next	Write	Query replace
CONT	Yank	Redisplay		
Line No.		Help		
Renumbering		SCRATCH		

図5-39 ポップアップメニュー一覧

各項目間の移動は<←>, <→> (編集キーボード) または<F6>~<F10> (ファンクション・キーボード) キーで行ないます。<←>, <→> キー (編集キーボード) は左右に移動し、<F6>~<F10> キー (ファンクション・キーボード) はそれぞれの項目に割り当たっています。

F6	BASICメニュー
F7	Regionメニュー
F8	Window/Otherメニュー
F9	Fileメニュー
F10	Search/Replaceメニュー

図5-40 ポップアップメニューに関する
ファンクション・キーボード

該当する項目のメニューが表示されたら、<←>, <→> キー (編集キーボード) で実行する機能を選択します。

Do (Ctrl-A)	ポップアップメニューの表示
F11 (Ctrl-Z)	ポップアップメニューの中止

図5-41 ポップアップメニューに関する
ファンクション&メイン・キーボード

メニュー	説明	参照先
MOVE and RUN BASIC mode RUN CONT Line no. Renumbering	BASICプログラムを転送し、実行する BASICコマンドを実行する すでに転送済みのプログラムを実行する Ctrl-Cにより停止したプログラムを継続する 行番号を設定する 行番号を再設定する	5.9.2項 5.9.4項 5.9.2項 5.9.5項 5.9.1項 5.9.1項
Set mark Kill region Copy region Yank	テキストの削除、コピーのための先頭位置をセットする テキストのマークからカーソル位置までを削除する テキストのマークからカーソル位置までを保存する 保存されたテキストをコピー（復旧）する	5.4.1項 5.4.1項 5.4.3項 5.4.2項
Only Split Next Redisplay Help SCRATCH	ウィンドウを一つにする ウィンドウを上下に分割する 他のウィンドウにカーソルを移動する テキストの再表示を行なう VG-920に割り当てられたエディタ機能を表示する エディタを初期化する	5.5.2項 5.5.1項 5.5.2項 5.5.3項 5.10節 5.13節
Load Save Write	メモリ・カードからファイルをロードする ファイルを更新する メモリ・カードへファイルをセーブする	5.6.2項 5.6.3項 5.6.1項
Forward search Backward search Query replace	カーソル位置から後方へ向かって文字列を検索する カーソル位置から前方へ向かって文字列を検索する 文字列の置き換えを行なう	5.7節 5.7節 5.8節

図5-42 ポップアップメニューの説明

5.12 エディタ機能の中止

実行中のエディタ機能を中止するときには、<F11> (ファンクション・キーボード) または <Ctrl-Z> キー (メイン・キーボード) を入力します。

注) BASICバッファへテキスト転送中や、ファイルのセーブ/ロード中に中止できません。

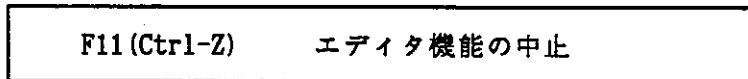


図5-43 キャンセルに関するファンクション (メイン) ・キーボード

5. 1 3 エディタの初期化

現在起動しているエディタを初期化します。<F12>キー（ファンクション・キーボード）またはポップアップメニューの「初期化」を入力すると、ミニウィンドウが表示され、編集中のテキストをセーブするかどうかを尋ねてきます。（テキストを変更した場合のみ）

それに返答すると、エディタを初期化するかどうかを尋ねてきます。

<Y> <Return>と入力すると、エディタは初期化され、<N> <Return>と入力すると初期化は中止し、元のエディタ画面に戻ります。

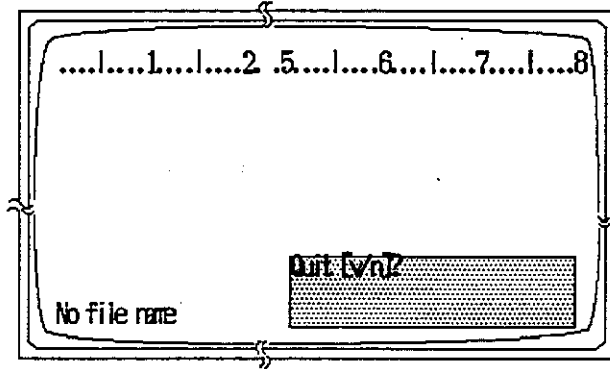


図5-44 エディタの終了ミニウィンドウ

↓ <Y> <Return> を入力する

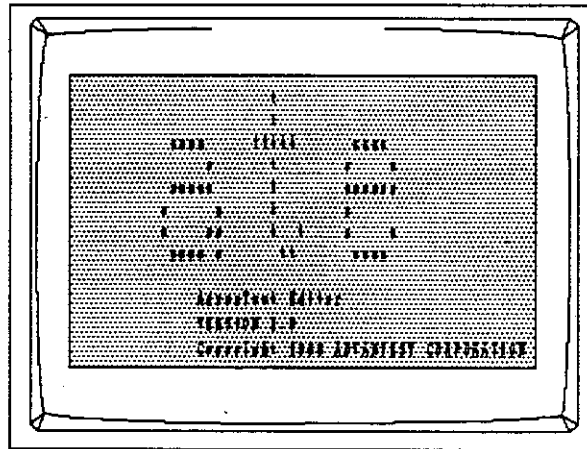


図5-45 エディタの初期化

F12	ateエディタを初期化する
-----	---------------

図5-46 エディタの初期化に関するファンクション・キーボード

6. 特長（計測器のみの場合）

ateエディタは、高度な編集機能を備えたフルスクリーンエディタです。ateエディタの機能には、カーソル制御、テキストの挿入、削除、コピー、移動、置換、検索、ウィンドウ、ファイル、BASICプログラムの実行、などがあります。また、これらの機能を円滑にするためにポップアップメニューがあります。

以下に外部端末を用いた場合との相違点を表します。

	計測器+VG-920	計測器のみ
最大カラム数	80	54
最大行数	23	17
キーボード	全機能の割当	一部機能の割当
文字入力	キーボードより入力	ラベル機能により入力

図6-1 外部端末使用時との相違点

6.1 キーボード

計測器に外部端末を接続した場合、全機能が端末のキーボードに割り当てられていたのに対し、計測器のみの場合、一部の機能だけが正面パネルに割り当てられています。

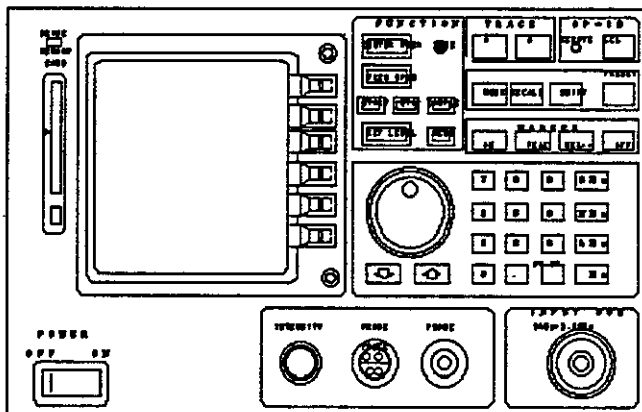


図6-2 R3261/3361正面パネル

6. 2 ポップアップメニュー

外部端末を使用したときと同様に、カーソル移動を除く全ての機能がポップアップメニューで実行します。キーボードがないときは、このポップアップメニューでの編集が主になります。

実行したい機能をポップアップメニュー内よりカーソルキーで選択し、<編集>キー (**GHZ** , **MHz** , **kHz** , **Hz**) を押すと、その機能が実行されます。

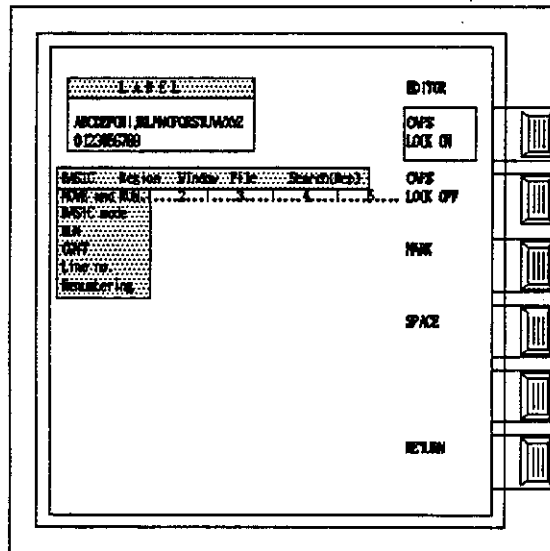


図6-3 ポップアップメニュー

6. 3 BASIC実行環境

ateエディタで編集したBASICプログラムをそのままエディタ上で実行できます。
BASICのRUN, CONT, SCRATCHコマンドは、ソフトキーまたはポップアップメニューで実行します。それ以外のコマンドはBASICモードのミニウィンドウ内で実行します。

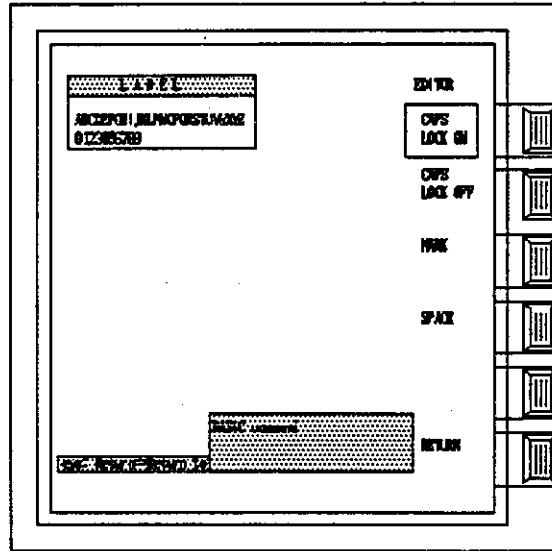


図6-4 BASICモード

6. 4 ラベルの定義

ateエディタでは、行番号は省略し、ラベルを定義します。これにより、サブルーチン・コールなどを使うとき、わかりやすいプログラムが作成できます。

注) ラベルは文字列の先頭に必ず* (アスタリスク) をつけます。

```
A=1:B=0
*LOOP
    B=B+A^2
    IF B>10000000 THEN GOTO *ENDLOOP
    PRINT B
    GOTO *LOOP
*ENDLOOP
STOP
```

図6-5 ラベルの定義

6. 5 ファイル

作成したプログラムは、ファイル単位で本体のメモリ・カードへセーブできます。また、メモリ・カードからファイルをロードして、編集もできます。

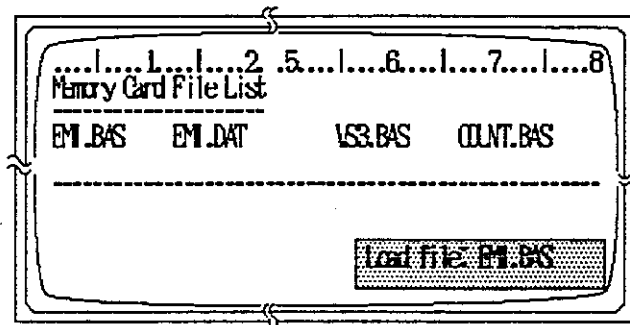


図6-6 ファイルのロード

6. 6 文字入力

計測器に外部端末を接続したときは、キーボードから入力できますが、計測器のみではラベル機能を用いて入力します。この時、<SHIFT>キー（計測器パネル）を押してからラベル機能の要領で文字を入力します。

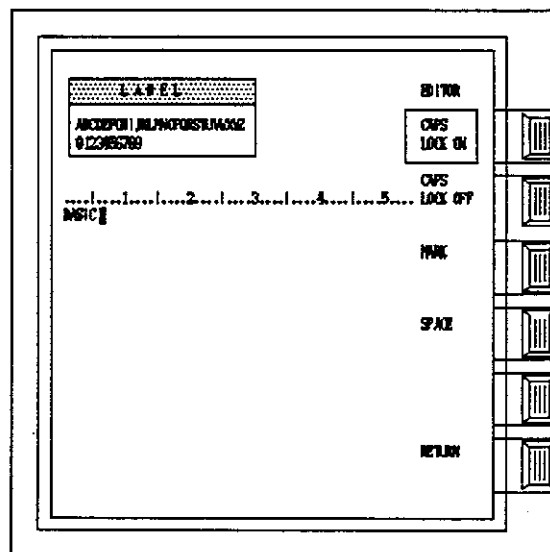


図6-7 文字入力

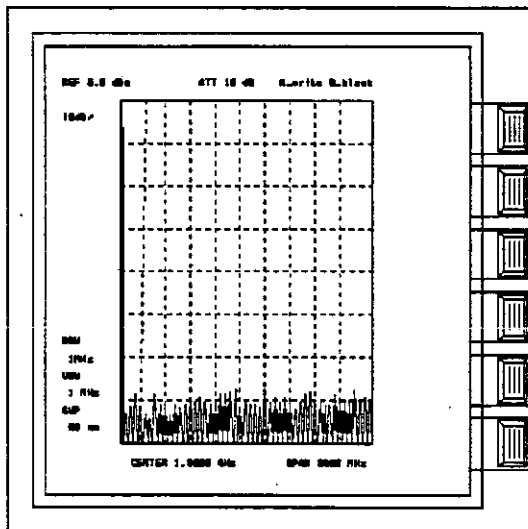
6. 7 ヘルプメニュー

計測器に外部端末(VG-920)を接続したときは、割り当てられている機能を一覧表示しますが、計測器のみではサポートしていません。

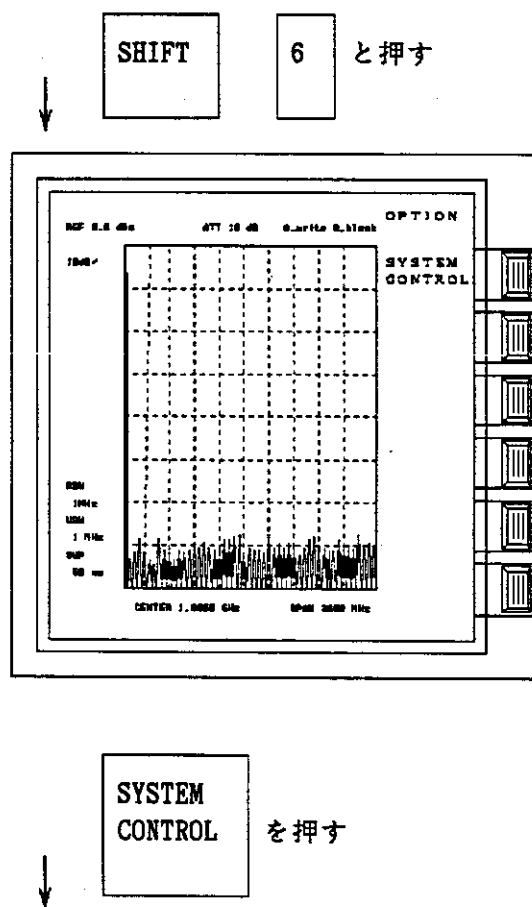
7. 起動 (計測器のみの場合)

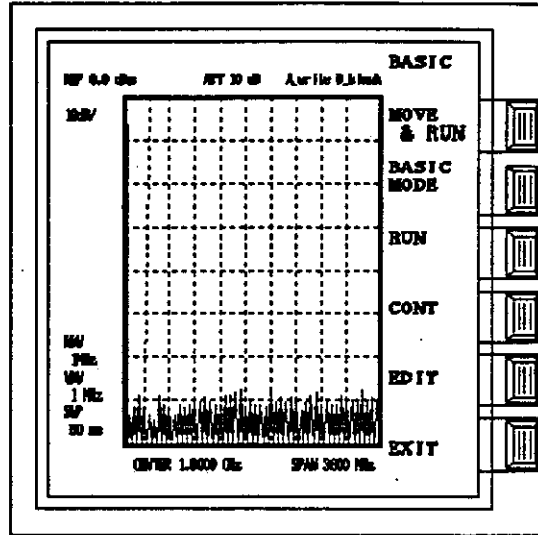
(1) 操作

計測器の電源を入れ、以下に示す操作をすると、初期画面が表示され、ateエディタが起動します。



計測器の正面パネル上 (計測器パネル) でキー操作を行ないます。





EDIT を押す
↓

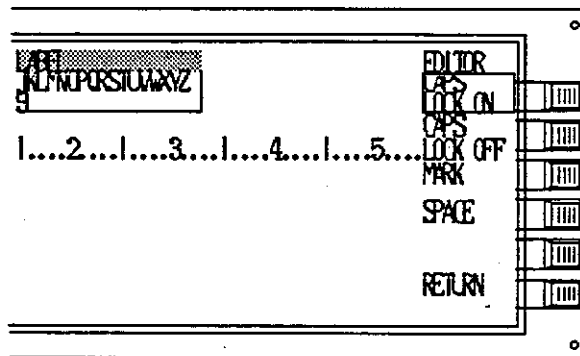


図7-1 ateの起動方法

(2) 画面表示の説明

画面の表示機能を説明します。

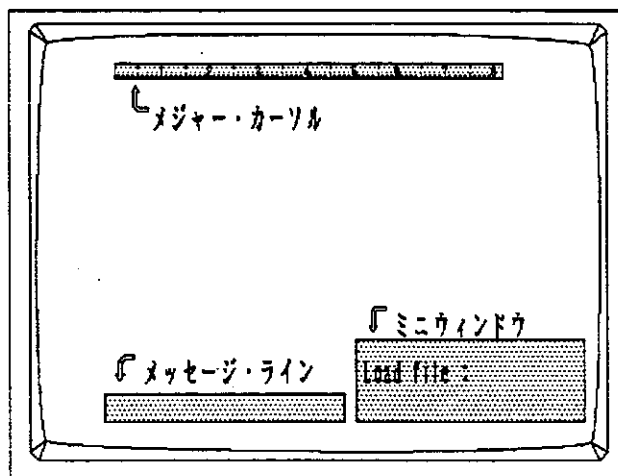


図7-2 ateの動作画面

- ・メジャーカーソル
編集時にカーソル位置を把握するためのメジャー表示です。
- ・メッセージライン
コマンドの実行手順、実行結果、エラーメッセージを表示します。
- ・ミニウインドウ
コマンドを実行する場合、パラメータが必要なときに使います。バッファ名、ファイル名、文字列などパラメータを要求するコマンドに応じて様々です。
表示されるプロンプトに従ってパラメータを入力します。パラメータの後に必ず<単位>キー ([GHZ] , [MHz] , [kHz] , [Hz]) を入力して下さい。プロンプトの後ろの [] で囲まれた部分はデフォルトパラメータです。このデフォルトはパラメータを入力しないで<単位>キー ([GHZ] , [MHz] , [kHz] , [Hz]) を押すとパラメータになります。

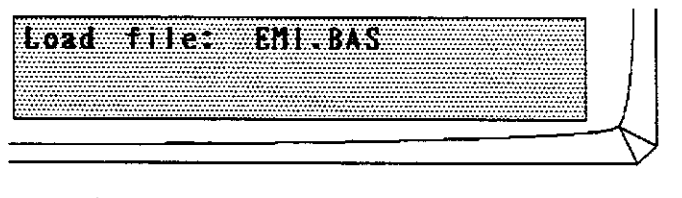


図7-3 ミニウインドウ

MEMO 

8. 機能一覧（計測器のみの場合）

キーボードがないため、エディタの各機能の実行はポップアップメニューによって行ないます。しかし、一部の機能は、以下に示すように本体の正面パネルに割り当てられています。

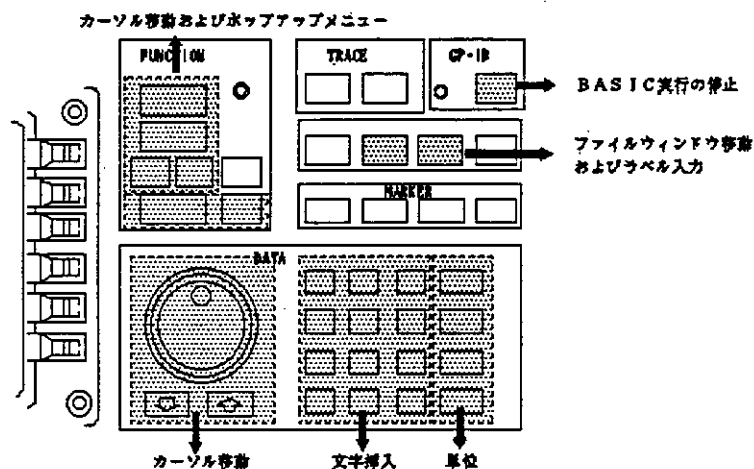


図8-1 計測器パネルのキー配置

8. 1 カーソル制御

◎ (→ノブ)	次の文字に移動
◎ (←ノブ)	前の文字に移動
↑	前の行に移動
↓	次の行に移動
STARTキー	ファイルの先頭に移動
STOPキー	ファイルの末尾に移動
SPANキー	前に一画面分移動
CENTERキー	次に一画面分移動

8. 2 挿入

ラベル機能	一般文字を挿入
数値キー	数字を挿入
単位キー	改行文字を挿入
SPACEキー	スペースを挿入
ピリオドキー	ピリオドを挿入

8. 3 削除

BK SPキー	カーソル前の文字を削除
---------	-------------

8. 4 コピー、移動、削除 (範囲指定)

Set Mark	リージョン処理に必要なマークをカーソル位置にセットする (ポップアップ)
Kill Region	マークからカーソル位置までを削除し、バッファに入れる (ポップアップ)
Copy Region	マークからカーソル位置までをバッファに入れる (ポップアップ)
Yank	<kill>または<copy>によりバッファに入れられた部分をカーソル位置にコピーする (ポップアップ)

8. 5 ウィンドウ

Only	ウィンドウを1つにする (ポップアップ)
Split	ウィンドウを2つに分割する (ポップアップ)
Next	カーソルを次のウィンドウに移動 (ポップアップ)
redisplay	画面をクリアして再表示する (ポップアップ)
COUPLEキー	画面をクリアして再表示する

8. 6 ファイル

Load	メモ리카ードから指定したファイルをロードする (ポップアップ)
Save	メモ리카ードにファイルをセーブする (ポップアップ)
Write	メモ리카ードに指定したファイル名でセーブする (ポップアップ)
RECALLキー	ファイル名一覧ウィンドウへカーソルを移動する

8. 7 検索

Forward search	前方に文字列を検索 (ポップアップ)
Backward search	後方に文字列を検索 (ポップアップ)

8. 8 置換

Query replace	文字列を置き換え (ポップアップ)
---------------	-------------------

8. 9 BASICモード

Line no.	自動的な行番号挿入を設定 (ポップアップ)
Renumbering	行番号の再定義 (ポップアップ)
MOVE and RUN	プログラムをBASICバッファに転送後、実行する (ポップアップ)
BASIC mode	BASICモードへ移行する (ポップアップ)
RUN	すでに転送済みのプログラムを実行する (ポップアップ)
CONT	<LOCAL>キーで中断したプログラムを続行する (ポップアップ)
MOVE & RUNキー	プログラムをBASICバッファに転送後、実行する
BASIC MODEキー	BASICモードへ移行する
RUNキー	すでに転送済みのプログラムを実行する
CONTキー	<LOCAL>キーで中断したプログラムを続行する

8. 10 ポップアップメニュー


REF LEVELキー ポップアップメニューを表示する

8. 11 キャンセル

MENUキー エディタの諸機能を中止する

8. 12 エディタの再起動

SCRATCH 現エディタを終了し、バッファを初期化して再起動する
(ポップアップ)

MEMO 

9. 機能解説 (計測器のみの場合)

9. 1 カーソル制御

カーソルは編集位置を示すものであり、編集はカーソルを基準にして行なわれます。

カーソル制御とは、画面上でカーソルを特定の場所に移動することをいいます。まず、1文字づつ上下左右に移動する機能があります。これらの機能はそれぞれカーソルキーに割り当てられており、現在の場所から新しい場所へカーソルを移動します。

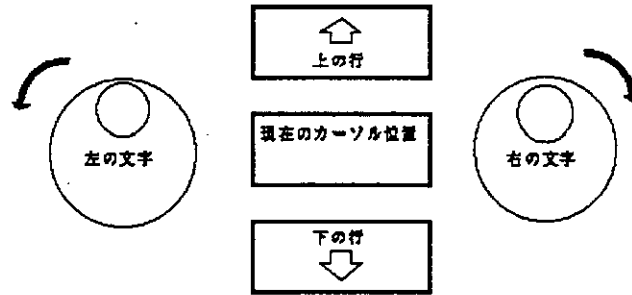


図9-1 カーソルキー

これらは基本的なカーソル移動機能を持ち、頻繁に使います。
画面の上限や下限を超えて移動しようとする時、その方向にあるテキストがスクロールし、カーソルは常に画面内に存在します。

この他に正面パネルに割り当てられている機能があります。

STARTキー	ファイルの先頭に移動する
STOPキー	ファイルの末尾に移動する
CENTERキー	次のページに進む
SPANキー	前のページに戻る

図9-2 カーソル移動に関する計測器パネル

9. 2 挿入

ateエディタでは、常にテキストの挿入ができる状態となっているので、ラベル機能と同様に文字を入力すると、すべて挿入されます。文字は一般文字と制御文字の2つに分けられます。一般文字とは'A', '1'など目に見える文字を言い、制御文字とは'ESC', 'CTRL'など目に見えない文字を言います。

一行が画面の幅以上に長くなると、右端に'\$'記号を表示して、その行が継続されていることを示します。

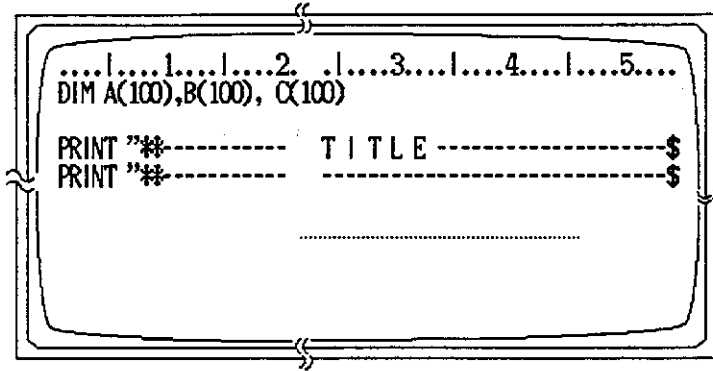


図9-3 文字の継続を表す

行の終わりは正面パネルの< 挿入 >キーを入力すると、改行文字で表わされます。

ラベル機能	一般文字を挿入
数値キー	数字を挿入 (計測器パネル)
単位キー	改行文字を挿入 (計測器パネル)
SPACEキー	スペースを挿入 (計測器ソフトメニュー・キー)
ピリオドキー	ピリオドを挿入 (計測器パネル)

図9-4 挿入に関するキーパッド

注意

通常、正面パネルはateエディタの機能が割当てられているので、文字の挿入はできません。端末のキーボードと同様に文字を挿入するには、< 挿入 >キー (計測器パネル)を入力してから、ラベル機能または数値キーなどで文字を挿入します。

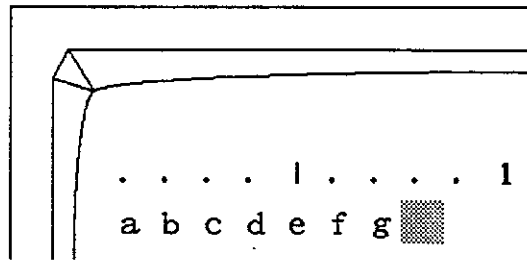
9.3 削除

削除には、一文字削除、一行削除、範囲指定削除があります。ここでは一文字削除を説明します。その他の削除方法は次項で説明します。

9.3.1 一文字削除

一文字削除とは、カーソル直前の文字を削除することを言います。

カーソル直前の文字を削除するには、正面パネルの<BK SP>キー（計測器パネル）を入力します。



↓ <BK SP>キーを押す

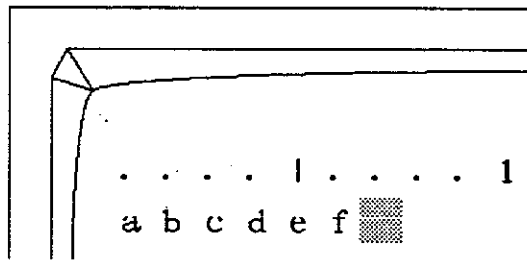


図9-5 カーソル直前の文字を削除

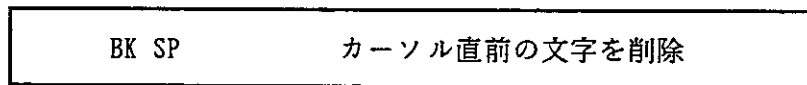


図9-6 削除に関する計測器パネル

9. 4 コピー、移動、削除（範囲指定）

コピー、移動、削除をするために範囲を指定して、テキストを保存します。

計測器パネルの<[F10] [F11]>キーを入力すると、画面上部にポップメニューが表示されます。

9. 4. 1 テキストの削除

(1) マークセット

削除、移動をするとき、削除する範囲の先頭位置にカーソルを移動して、ポップアップメニューの“Mark set”でマークを設定します。

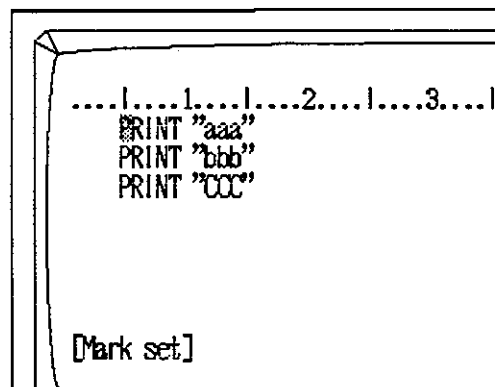


図9-7 マークセット

(2) テキストの削除

削除する範囲の最終位置よりも1文字分先へカーソルを移動し、ポップアップメニューの“Kill region”を入力します。すると、(1)でマークをセットした位置から現在のカーソルの手前までのテキストを削除します。

削除したテキストは、内部バッファに保存されているので、後述のテキストの移動、復旧、コピーなどは(1)、(2)の操作をした後、実行します。

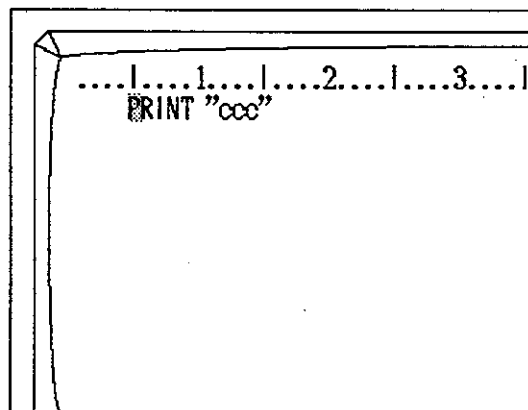


図9-8 テキストの保存（削除）

9. 4. 2 テキストの復旧、移動

削除したテキストを元に戻すには、現在のカーソル位置でポップアップメニューの"**undo**"を入力します。テキストを移動する場合は、移動したい位置へカーソルを移動し、同様にポップアップメニューの"**undo**"を入力します。

このようにポップアップメニューの"**undo**"は、保存したテキストを現在のカーソル位置に挿入する機能です。

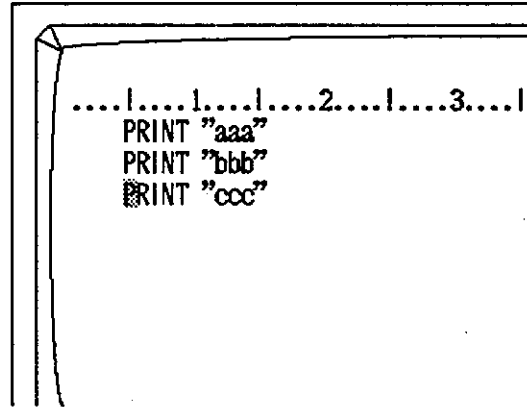


図9-9 テキストの復旧

9. 4. 3 テキストのコピー

テキストをコピーするには、範囲の先頭位置を削除する場合と同様に設定し、最終位置にカーソルを移動し、ポップアップメニューの"**copy region**"を入力します。すると範囲内は何も変わったように見えませんが、指定したテキストが内部バッファにセーブされているので、コピー位置にカーソルを移動し、ポップアップメニューの"**undo**"を入力します。

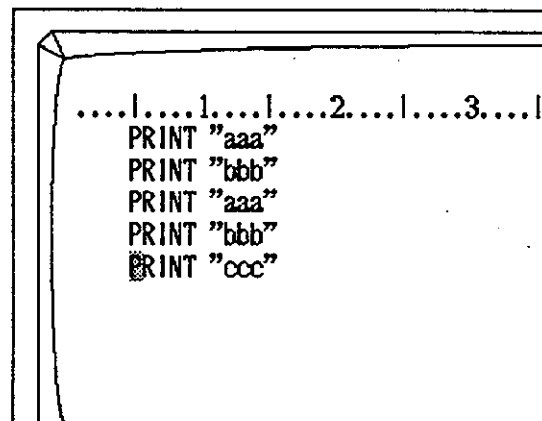



図9-10 テキストのコピー

ポップアップメニューの「」によって復旧されるテキストは、内部バッファ内の最新のテキストになります。

Set mark	範囲指定に必要なマークをカーソル位置にセットする
Kill region	マークからカーソル位置までを削除し、テキストを保存する
Copy region	マークからカーソル位置までのテキストを保存する
Yank	保存したテキストをカーソル位置に挿入する

図9-11 リージョン処理に関するポップアップメニュー

9. 5 ウィンドウ

通常のウィンドウはオーバーラッピング型とタイル型に分けられますが、ateエディタのウィンドウは画面を上下に分割するタイル型になります。

通常のウィンドウは1つですが、ateエディタは2つのウィンドウを持つことができ、それぞれに対してテキストを表示します。例えば、画面内に表示しきれない2つの部分を同時に参照できます。

注) オープンできるファイルは1つに限られているため、2つのウィンドウにそれぞれ別のファイルをオープンできません。

9. 5. 1 ウィンドウの分割

ウィンドウを2つに分割するには、ポップアップメニューの"**Split**"を入力します。

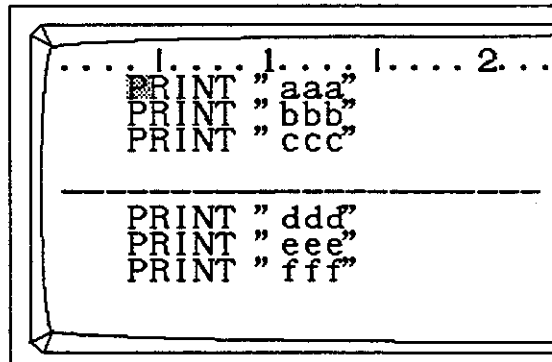


図9-12 上下に分割されたウィンドウ

9. 5. 2 ウィンドウの初期化

上下に分割されたウィンドウを1つに戻すには、ポップアップメニューの"**Quit**"を入力して、残したいウィンドウにカーソルを移動します。次に、ポップアップメニューの"**Quit**"を入力すると、ウィンドウは1つに戻ります。

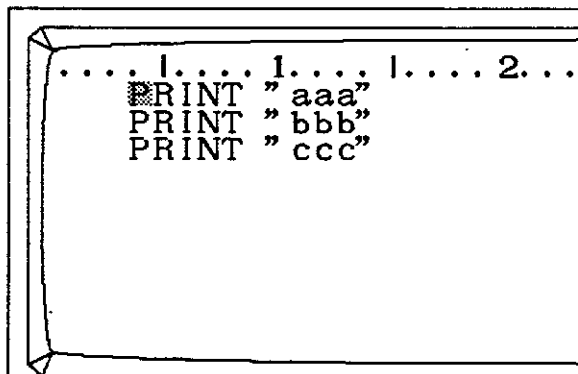


図9-13 ウィンドウを1つにする

9. 5. 3 画面の再表示

編集中のテキストを再表示したいときは、計測器パネルの<COUPLE>キーまたはポップアップメニューの"Redisplay"を入力して、テキストを再表示します。

Only	他のウィンドウを消去して、カーソルのあるウィンドウ1つにする
Split	ウィンドウを上下に分割し、カーソルを上ウィンドウに置く
Next	カーソルを次のウィンドウに移動する
Redisplay	画面をクリアして、テキストの再表示を行なう

図9-14 ウィンドウに関するポップアップメニュー

COUPLEキー	画面をクリアして、テキストの再表示を行なう
----------	-----------------------

図9-15 ウィンドウに関する計測器パネル

9.6 ファイル

ateエディタでは、テキストをファイルという単位として、本体のメモリ・カードへのセーブ、メモリ・カードからのロードができます。編集したテキストは必ずメモリ・カードへセーブするようにします。

注) セーブしないままエディタを終了すると、編集したテキストは失われます。

ここでは、ファイルのセーブ/ロードを説明します。

計測器パネルの<REF LEVEL>キーを入力すると、画面上部にポップアップメニューが表示されます。

9.6.1 ファイルのセーブ

編集したテキストをメモリ・カードにセーブするには、ポップアップメニューの"Write"を入力します。ここでセーブするファイルに名前をつけます。ファイル名は最大10文字です。セーブが終了すると、書き込んだ行数をメッセージラインに表示します。

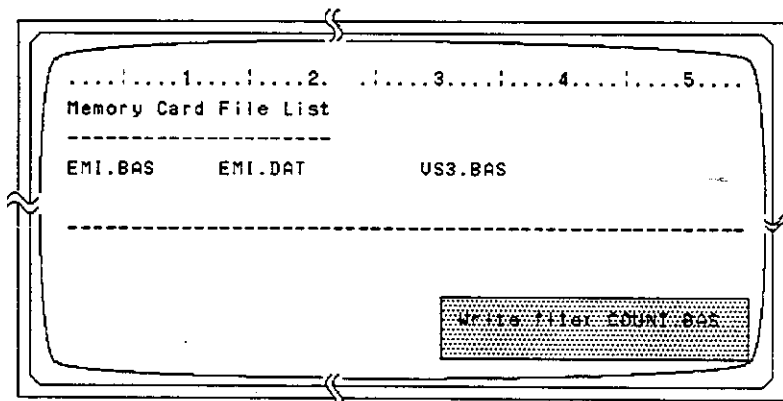


図9-16 テキストのセーブ (ファイル名を入力)

9. 6. 2 ファイルのロード

セーブされているファイルをロードするには、ポップアップメニューの"**Load**"を入力します。すると、画面上に現在メモリ・カードにセーブされているファイルが一覧表示され、どのファイルをロードするかを尋ねてきます。ここでファイル名を入力するか、または一覧表示されているウィンドウへ正面パネルの<**MEMO**>キー（計測器パネル）を押してカーソルを移動させ、その後、カーソルキーまたはデータ・ノブにてファイル名を選択し、最後に **[Hz]** キー（計測器パネルの単位を表わすキー）を入力します。

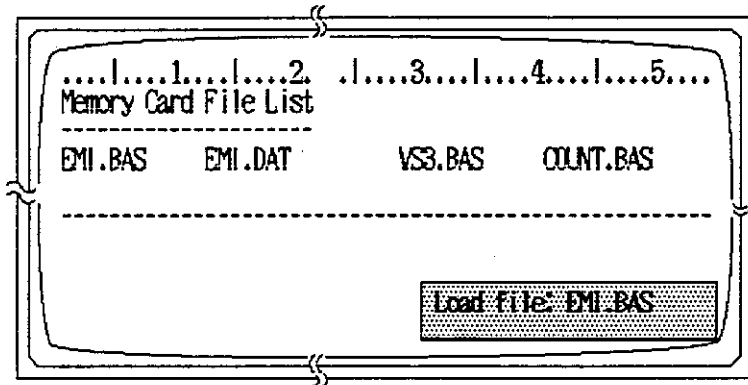


図9-17 ファイル名の入力

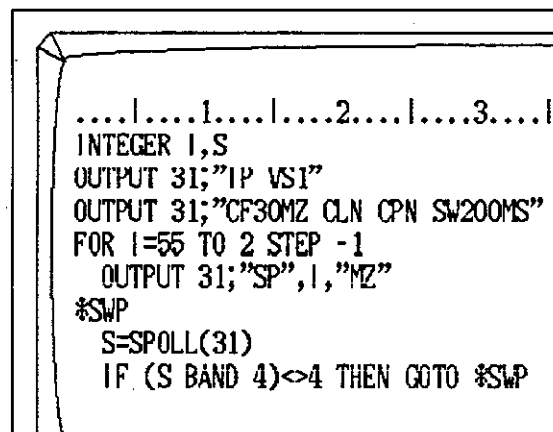


図9-18 ファイルのロード

メモリ・カードからファイルをロードしている最中は、本体のメモリ・カード挿入口上部のLEDが点灯します。ロードが終了すると、画面上にテキストが表示され、メッセージラインに読み込んだ行数が表示されます。

この時、行番号がついているものは自動行番号挿入(AUTO)機能が設定されます。

また、新しいファイルを作成するときは、そのファイル名を入力します。

メモリ・カードが挿入されていないときは、ブザー音またはエラーメッセージが表示されます。

9.6.3 ファイルの更新

同じファイル名でセーブする場合には、ポップアップメニューの"Save"を入力します。すると、セーブするかどうかを尋ねてきます。テキストに何の変更もない場合は"No change"というメッセージを表示します。

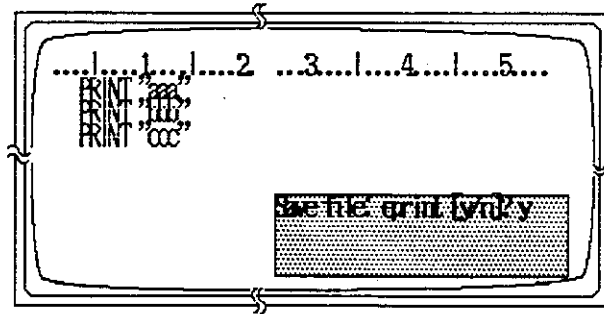


図9-19 テキストの更新 (同一ファイル名)

ラベル機能でyと入力し、**Hz** キー (計測器パネルの単位を表わすキー) を押すと、テキストはファイルに書き出されます。セーブが終了すると、そのファイルの行数がメッセージラインに表示されます。

ラベル機能でnと入力し、**Hz** キー (計測器パネルの単位を表わすキー) を押すと、テキストのセーブは実行しません。

Load	メモリ・カードから指定したファイルをロードする
Save	メモリ・カードに同一ファイル名でセーブする
Write	メモリ・カードに指定したファイル名でセーブする

図9-20 ファイルに関するポップアップメニュー

RECALLキー	ファイル名一覧ウィンドウへカーソルを移動する
----------	------------------------

図9-21 ファイルに関する計測器パネル

REF LEVELキー	ポップアップメニューの表示
MENUキー	ポップアップメニューの中止

図9-22 ポップアップメニューに関する計測器パネル

◎ (ノブ)	
↑ キー	カーソルの移動
↓ キー	

図9-23 カーソルに関する計測器パネル

GHzキー	
MHzキー	キー入力の終了 (ENTER)
kHzキー	
Hzキー	

図9-24 キー入力に関する計測器パネル

9.6.4 ファイルの消去

第2部 システム・コントローラの2章 コマンドとステートメントの文法と解説
(58) PURGEを参照して下さい。

9.6.5 プログラム・オート・スタート機能

電源オン時にプログラムを自動的にメモリ・カード(読み込み)、MOVE and RUN(起動)させることができます。

作成したプログラムを“AUTOSTART”というファイル名で、メモリカードに登録(SAVE)します。

ポップアップメニューの[Write]で、ファイル名AUTOSTARTとしてメモリカードに登録します。

```
Write File: AUTOSTART
```

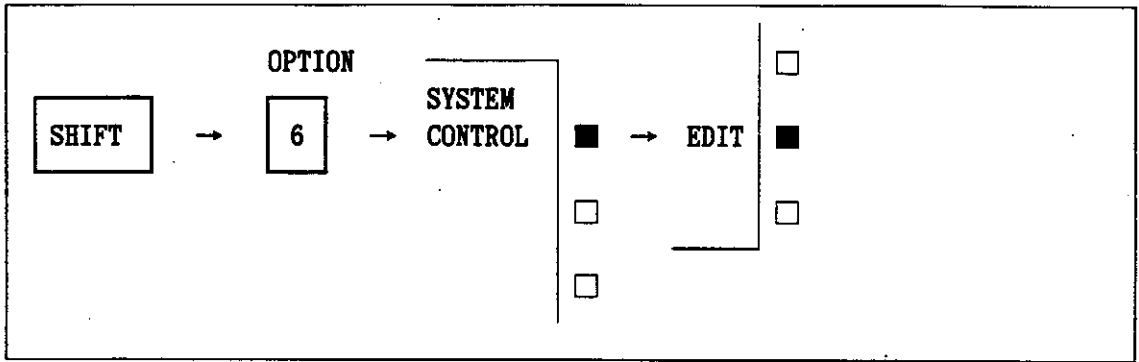
CAT命令で確認

```
Card:032k      Page:02/02
File:023/024  MAXbyte:025600
NO. + File-Name + Type      + Byte
013+ AUTOSTART + P          + 001055
014+           +            + 000000
015+           +            + 000000
```

次回電源オン時からファイル名“AUTOSTART”を探し、AUTOSTARTが存在したら、そのファイルをロードして起動します。

注意

1. AUTOSTART のファイル名は必ず大文字で登録して下さい。
AUTOSTART はメモリーカード1枚につき、1ファイルしか存在できません。
AUTOSTART の起動は電源オン後、1回のみ有効です。
2. 電源オン時にRS232C端子にターミナル等が接続されている場合は、そのままプログラムロードし実行しますが、RS232C端子になにも接続されていない場合は、以下の操作でAUTOSTARTが起動します。



ターミナルを接続しない状態で起動させる操作

9.7 検索

文字列をテキスト内で前方または後方に向かって探す機能です。

計測器パネルの<REF LEVEL>キーを入力すると、画面上部にポップアップメニューが表示されます。

カーソル位置以降を検索するには、ポップアップメニューの"Forward search"を入力します。カーソル位置以前を検索するには、ポップアップメニューの"Backward search"を入力します。両者は検索の方向が異なるだけですべて同じ働きをします。

以下にカーソル位置以降の文字列の検索手順を示します。

ポップアップメニューの"Forward search"を入力します。画面にミニウィンドウが表示され、検索する文字列を尋ねてきたら、探したい文字列を入力します。

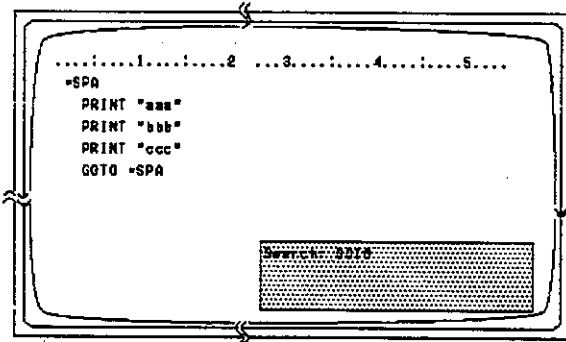


図9-25 カーソル位置以降の文字列検索

検索した文字列の先頭にカーソルが移動し、検索は終了します。同じ文字列を続けて検索したいときは、同様の操作を行いますが、文字列の入力は省略できます。

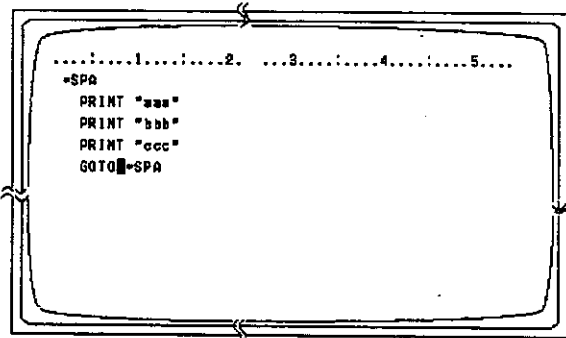


図9-26 実行結果

検索できないときは、"Not found"というメッセージが表示されます。

Forward search	カーソル位置以降に向かって検索する
Backward search	カーソル位置以前に向かって検索する

図9-27 検索に関するポップアップメニュー

9.8 置換

カーソル位置以降のテキスト中の任意の文字列を変更する機能です。

計測器<PF LEVEL>キーを入力すると、画面上部にポップアップメニューが表示されます。

それでは、文字列を置き換えてみます。ポップアップメニューの"Query replace"を入力します。画面にミニウィンドウが表示され、置換したい文字列を尋ねてきたら、対象となる文字列を入力します。

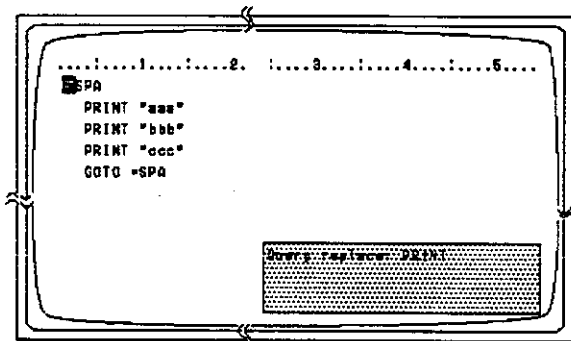


図9-28 置換対象文字列の入力

次に新たに変更する文字列を尋ねてきたら、任意の文字列を入力します。

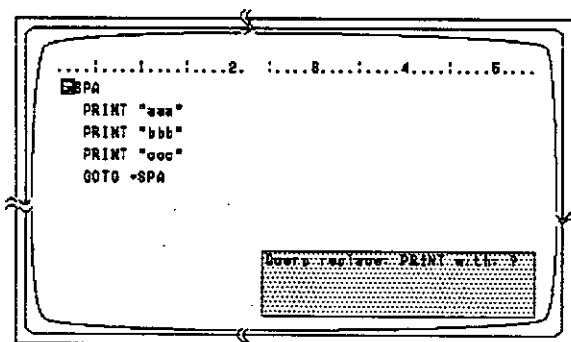


図9-29 置換文字列の入力

該当する文字列を検索すると、カーソルがその文字列の先頭に移動し、置換するかどうか待ち状態になります。ここで置換するときにはソフトキーの<SPACE>キー（計測器パネル）を入力します。置換しないときには正面パネルの<BK SP>キー（計測器パネル）を入力します。また、置換して終了するときには正面パネルの<>キー（計測器パネル）を入力します。置換しないで終了するときには正面パネルの<MENU>キー（計測器パネル）を入力します。

```
.....1.....2. ....3.....4.....5....  
=SPA  
? "aaa"  
PRINT "bbb"  
? "ccc"  
GOTO =SPA
```

図9-30 実行結果

Query replace 文字列を置き換える

図9-31 置換に関するポップアップメニュー

SPACEキー	置換する
BK SPキー	置換しない
ピリオドキー	置換して、終了する
MENUキー	終了する

図9-32 置換に関する計測器パネル

9.9 BASIC

ateエディタは、計測器上でのBASICプログラミングを行なうために開発されたポータブルエディタです。よって編集しながらBASICの実行、デバッグが容易にできます。その他にBASICモードに移行してから様々なBASICコマンドが実行できます。

計測器パネルの<REF LEVEL>キーを入力すると、画面上部にポップアップメニューが表示されます。

ここではエディタ上から実行できるBASICコマンドを説明します。

9.9.1 行番号の設定

今までBASICプログラムを作成するときには行番号を使いました。ateエディタでは行番号の代わりにラベルを用いて、プログラムを分かりやすくします。

```
A=1:B=0
*LOOP
      B=B+A^2
      IF B>10000000 THEN GOTO *ENDLOOP
      PRINT B
      GOTO *LOOP
*ENDLOOP
STOP
```

図9-33 ラベルを用いたプログラム

また、自動行番号挿入(AUTO)機能により行番号をつけながら編集もできます。ポップアップメニューの"Line no."を入力すると、ミニウィンドウが表示され、開始番号と間隔を指定します。これで編集時に [Hz] キー（計測器パネルの単位を表わすキー）を押すと、行番号が自動的に出力されます。

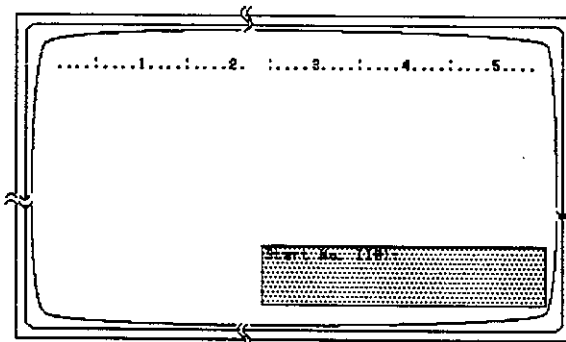


図9-34 開始番号の指定

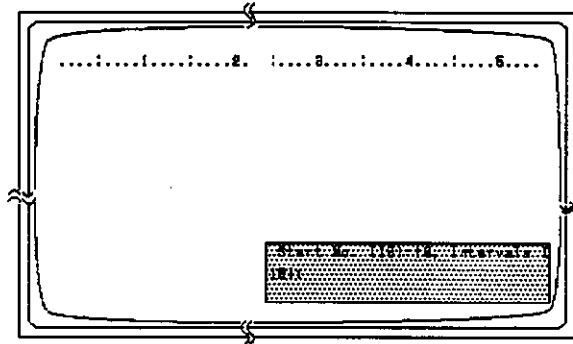


図9-35 行番号間隔の指定

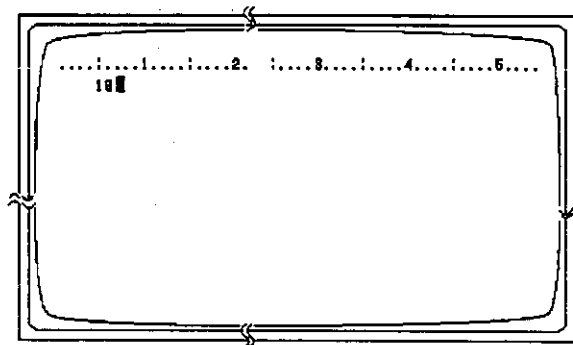


図9-36 自動行番号機能

また、編集途中の場合でも、この機能により行番号を追加することができます。行間での挿入を行なったときは、上の行番号にプラス1した数が行番号となります。

注) 一度この機能を実行すると、エディタの初期化 (9.13節) をしない限り継続されます。

自動行番号挿入機能と同様、行番号のリナンバリング機能があります。これは行番号を一定の指定にしたがって付け直すものです。ポップアップメニューの"Renumbering"を入力すると、自動行番号挿入機能と同様に、開始番号と間隔を指定されます。

9.9.2 BASICの実行

プログラムを実行する前に、ateエディタとBASICインタプリタの関係を説明します。

プログラムの編集はateエディタで制御し、完成したプログラムはateエディタの内部バッファに格納されます。これとは別にBASICインタプリタも内部バッファを持っていて、実行するためにはこのバッファにBASICプログラムが存在しなければなりません。

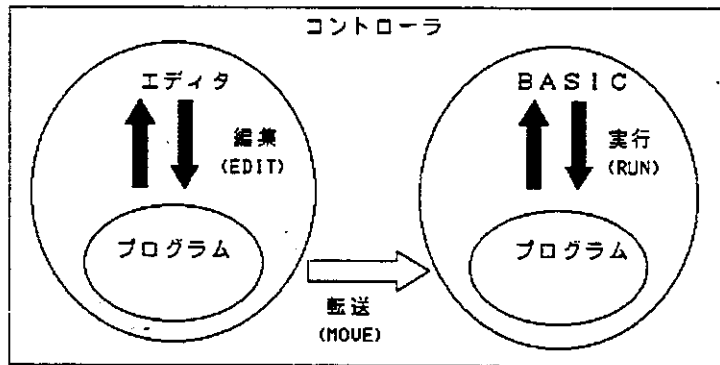


図9-37 エディタとインタプリタとの関係

よってBASICプログラムを実行する前に、エディタからインタプリタにプログラムを転送します。ソフトキーの<MOVE & RUN>* キー、またはポップアップメニューの"MOVE and RUN"を入力すると、エディタからプログラムが転送され、実行します。この時、前回実行したプログラムは消去されます。また、実行中にエラーが起こると、エディタに戻り、カーソルがエラー行に移動します。

すでに転送済みのプログラムを実行するときは、ソフトキーの<RUN>* キー、またはポップアップメニューの"RUN"を入力します。プログラムを転送する前に、上記の手順で実行したときは、"Program is not exist"というエラーメッセージが表示されます。

* MOVE&RUN, RUNは、 と押すと、画面右側に表示されます。

9.9.3 BASICの停止

実行中のBASICを停止するには、<LCL>キー（計測器パネル）を入力します。

9.9.4 BASICモード

BASICモードとは、BASICに対してのコマンドを実行できる環境を言います。ソフトキーの<BASIC MODE>キー、またはポップアップメニューの"BASIC mode"を入力すると、ミニウィンドウが表示されます。

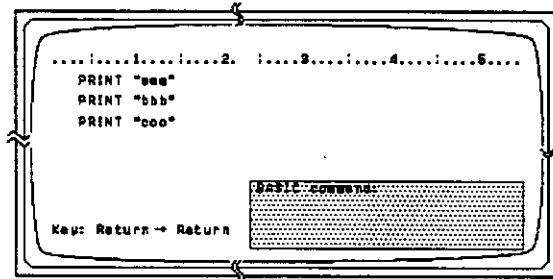


図9-38 BASICモード

実行できるコマンドは、RUN, LIST, CONTなどです。(コマンドの説明は [第2部 システム・コントローラ] を参照)

注) プログラムの内容を変更する場合は必ずエディタに戻ってから行ないます。

9.9.5 BASICの継続

<LCL>キー (計測パネル) またはPAUSE命令で一時停止したプログラムを続けて実行するときは、<CONT>キー (計測器ソフトメニュー・キー)、またはポップアップメニューの"CONT"を入力します。これはBASICコマンドの"CONT"と同じです。停止した次の文から実行し、実行するプログラムがない場合は、"Program cannot be continued"というエラーメッセージが表示されます。

Line no.	自動的に行番号を挿入する
Renumbering	行番号のリナンバリング
MOVE and RUN	プログラムを転送後、実行する
BASIC mode	BASICモードへ移行する
RUN	すでに転送済みのプログラムを実行する
CONT	<LOCAL>キーまたはPAUSE命令で中断したプログラムを続行する

図9-39 BASICに関するポップアップメニュー

MOVE & RUNキー	プログラムを転送後、実行する
BASIC MODEキー	BASICモードへ移行する
RUNキー	すでに転送済みのプログラムを実行する
CONTキー	<LOCAL>キーまたはPAUSE命令で中断したプログラムを続行する

図9-40 BASICに関する計測器パネル&計測器ソフトメニュー

9.10 ヘルプ

エディタの機能とキーボード (VG-920) の関係は画面上で確認できますが、正面パネルについてのヘルプメニューはサポートしていません。

9.11 ポップアップメニュー

ateエディタの機能は、カーソル移動機能を除いて、ほとんどがポップアップメニューによって実行できます。

カーソルキー以外のエディタ機能は、ほとんどポップアップメニュー内に表示され、その中から選択し、実行します。

計測器パネルの<REF LEVEL>キーを入力すると、画面上部にポップアップメニューが表示されます。

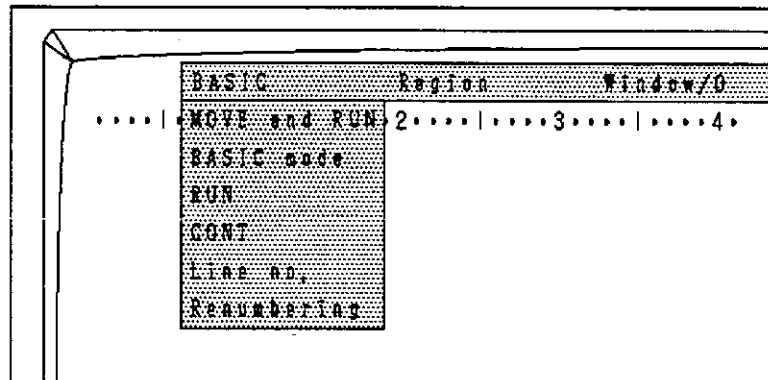


図9-41 ポップアップメニュー

BASIC	Region	Window/Other	File	Search/Replace
MOVE and RUN	Set mark	Only	Load	Forward search
BASIC mode	Kill region	Split	Save	Backward search
RUN	Copy region	Next	Write	Query replace
CONT	Yank	Redisplay		
Line No.		Help		
Renumbering		SCRATCH		

図9-42 ポップアップメニュー一覧

各項目間の移動は計測器パネルの<REF LEVEL>キーを続けて入力すると、トグルとなって移動します。

該当する項目のメニューが表示されたら、計測器パネルの<←>、<↓>キーで実行する機能を選択します。

REF LEVELキー	ポップアップメニューの表示
MENUキー	ポップアップメニューの中止

図9-43 ポップアップメニューに関する計測器パネル

メニュー	説明	参照先
MOVE and RUN BASIC mode RUN CONT Line no. Renumbering	BASICプログラムを転送、実行する BASICコマンドを実行する すでに転送済みのプログラムを実行する <LOCAL>キーにより停止したプログラムを継続する 行番号を設定する 行番号を再設定する	9.9.2項 9.9.4項 9.9.2項 9.9.5項 9.9.1項 9.9.1項
Set mark Kill region Copy region Yank	テキストの削除、コピーのための先頭位置をセットする テキストのマークからカーソル位置までを削除する テキストのマークからカーソル位置までを保存する 保存されたテキストをコピー（復旧）する	9.4.1項 9.4.1項 9.4.3項 9.4.2項
Only Split Next Redisplay Help SCRATCH	ウィンドウを1つにする ウィンドウを上下に分割する 他のウィンドウにカーソルを移動する テキストの再表示を行なう （サポートしていません） エディタを初期化する	9.5.2項 9.5.1項 9.5.2項 9.5.3項 9.10節 9.13節
Load Save Write	メモリ・カードからファイルをロードする ファイルを更新する メモリ・カードへファイルをセーブする	9.6.2項 9.6.3項 9.6.1項
Forward search Backward search Query replace	カーソル位置から後方へ向かって文字列を検索する カーソル位置から前方へ向かって文字列を検索する 文字列の置き換えを行なう	9.7 節 9.7 節 9.8 節

図9-44 ポップアップメニューの説明

9.12 エディタ機能の中止

実行中のエディタ機能を中止するときには、計測器パネルの<MENU>キーを入力します。

注) BASICバッファへテキスト転送中や、ファイルのセーブ/ロード中は中止できません。

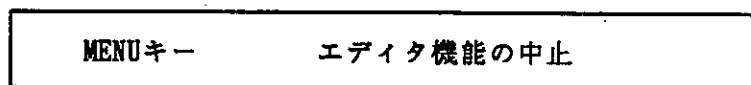


図9-45 キャンセルに関する計測器パネル

9.13 エディタの初期化

現在起動しているエディタを初期化します。ポップアップメニューの"SCRATCH"を入力すると、ミニウインドウが表示され、編集中のテキストをセーブするかどうかを尋ねてきます。(テキストを変更した場合のみ)

それに返答すると、エディタを初期化するかどうかを尋ねてきます。

ラベル機能で y と入力し、[Hz] キー(計測器パネルの単位を表わすキー)を押すと、エディタは初期化され、ラベル機能で n を入力し、[Hz] キー(計測器パネルの単位を表わすキー)を押すとと初期化は中止し、元のエディタ画面に戻ります。

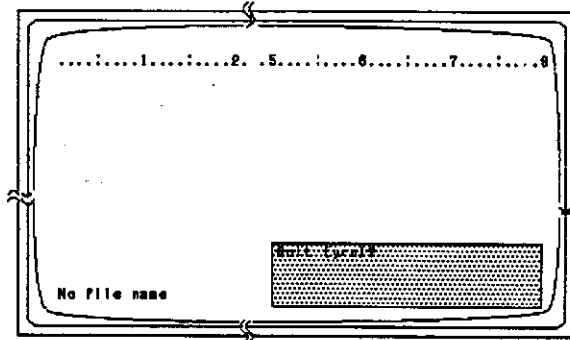


図9-46 エディタの終了ミニウインドウ



<y> <単位>キーを入力します。

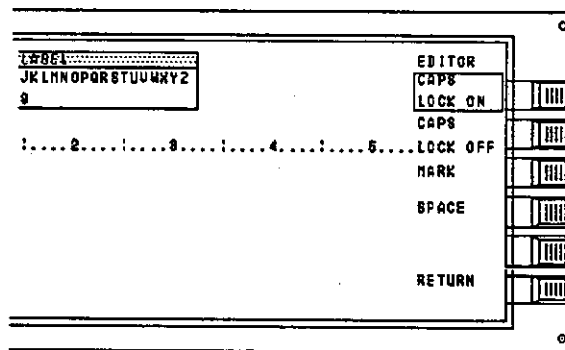


図9-47 エディタの初期化

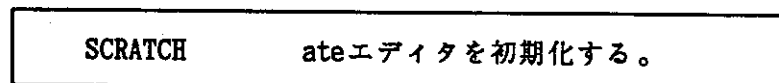
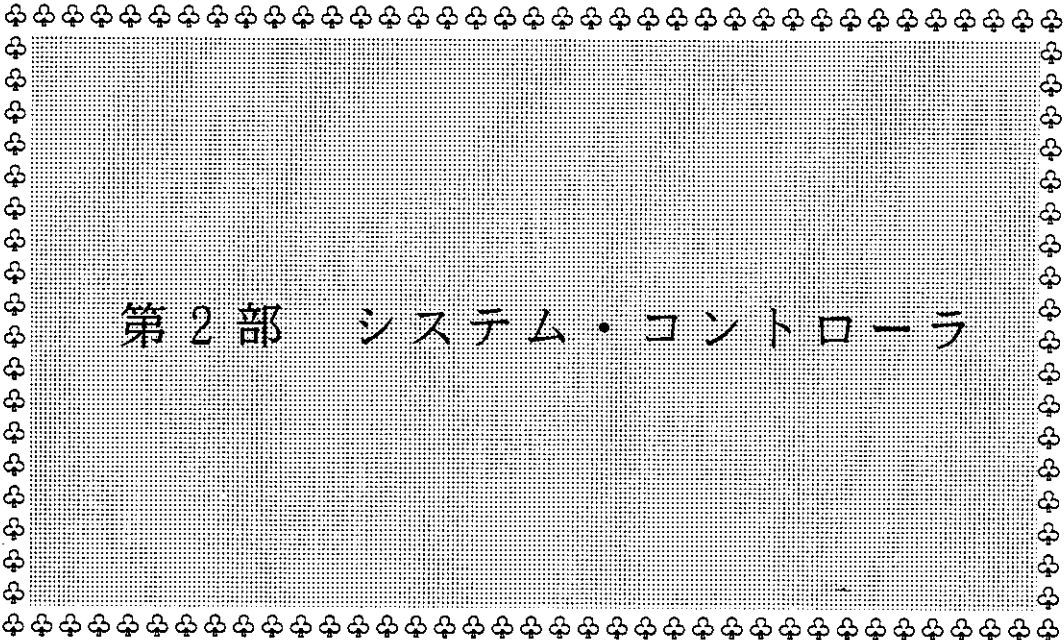


図9-48 エディタの初期化に関するポップアップメニュー

MEMO 



第 2 部 システム・コントローラ

第 2 部の目次

1. オプション 81 (BASIC GPIBコントローラ)	1- 1
1.1 概要	1- 1
1.2 BASICプログラミング	1- 2
1.2.1 プログラム構造	1- 2
1.2.2 キーワード	1- 3
1.2.3 ショート・ネーム	1- 3
1.3 オブジェクト	1- 4
1.4 定数	1- 5
1.4.1 整定数	1- 5
1.4.2 実定数	1- 5
1.4.3 文字列定数	1- 6
1.4.4 ラベル定数	1- 6
1.5 変数	1- 7
1.5.1 スカラ変数	1- 7
1.5.2 システム変数	1- 8
1.5.3 配列	1- 9
1.6 関数	1-10
1.6.1 組み込み関数	1-10
1.6.2 ビルトイン関数	1-12
1.7 演算式	1-14
1.7.1 代入演算子	1-14
1.7.2 単項算術演算子	1-16
1.7.3 2項算術演算子	1-18
1.7.4 論理演算子	1-19
1.7.5 比較演算子	1-20
1.7.6 サブ・ストリング演算子	1-21
1.7.7 ビット演算子	1-22
1.8 演算の優先順位	1-23
1.9 文字列演算	1-24
1.9.1 文字列の連結	1-24
1.9.2 文字列の比較	1-24
1.9.3 型変換	1-24
2. コマンドとステートメントの文法と解説	2- 1
2.1 概要	2- 1
2.2 はじめに	2- 1
2.2.1 説明の構成	2- 1
2.3 各種コマンドとステートメントの説明	2- 6
3. ビルトイン関数	3- 1
3.1 概要	3- 1
3.2 はじめに	3- 1
3.2.1 使用開始の前に	3- 1
3.2.2 使用上の注意	3- 4
3.2.3 説明の構成	3- 5
3.3 各種ビルトイン関数の説明	3-11
3.4 各種グラフィックの説明	3-56

1. オプション81 (BASIC GPIBコントローラ)

1.1 概要

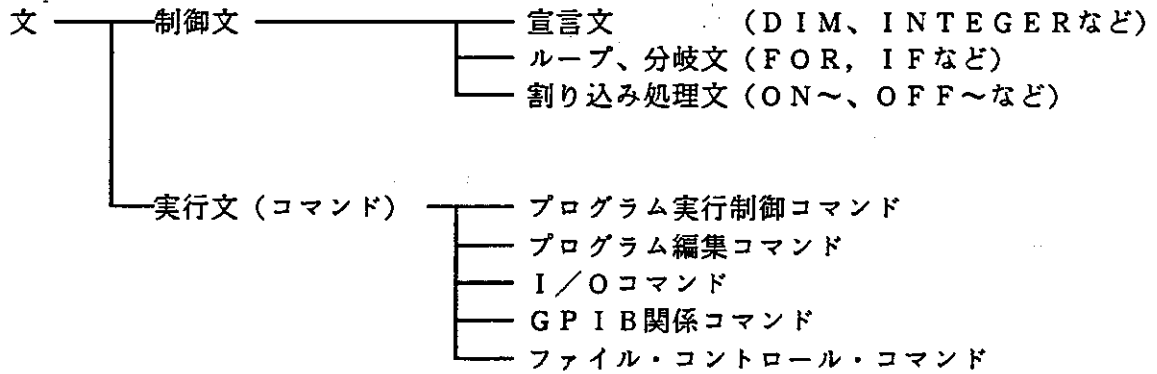
オプション81のBASIC言語は、汎用のBASICコマンドのほか、GPIB制御用コマンドを備えており、小規模GPIBシステムを構築でき、測定専用ビルトイン関数を使うと測定を簡単に、かつ高速に処理できます。

1.2 BASICプログラミング

1.2.1 プログラム構造

文は、BASICが処理する最小単位です。その文の集まりがBASICプログラムになります。

文は大きく分けて、制御文と実行文（コマンド）で構成されています。



各文は、キーワードと式から構成されます。この構成を取り決めたものが文法の構文規則です。

このBASICで予め、その意味およびその用途を定めている言葉をキーワードと言います。変数名などと同じ名前では使えません。

1.2.2 キーワード

【キーワード一覧】

AND	AS	ASCII	BAND	BINARY	BNOT
BOR	BREAK	BUZZER	BXOR	CASE	CAT
CLEAR	CLOSE	CLS	CMD	CONT	CONTINUE
CONTROL	CSR	CURSOR	DATA	DELIMITER	DIM
DISABLE	ELSE	ENABLE	END	ENT	ENTER
ERROR	FOR	GLIST	GLISTN	GOSUB	GOTO
GPRINT	IF	INIT	INITIALIZE	INP	INPUT
INTEGER	INTERFACE	INTR	ISRQ	KEY	LISTEN
LISTN	LLIST	LLISTN	LOCAL	LOCKOUT	LPRINT
NEXT	NOT	OFF	ON	OPEN	OR
OUT	OUTPUT	PAUSE	PRF	PRINT	PRINTER
PRINTF	PURGE	READ	REM	REMOTE	RENAME
REQUEST	RESTORE	RETURN	RUN	SCRATCH	SELECT
SEND	SPRINTF	SRQ	STEP	STOP	TALK
TEXT	THEN	TO	TRIGGER	UNL	UNT
USE	USING	WAIT	XOR		
実行できないキーワード群					
APPEND	BASIC	CHKDSK	COPY	COPYFILES	COUNT
DEL	DSTAT	ENTERF	FORMAT	LABEL	LOAD
MERGE	NEWVERSION	REN	SAVE	SYSTEM	TIME
POKE	DUMP				

注) 実行できないキーワードとは、キーワードとしては登録してあるが、命令の実行はできないものです。

1.2.3 ショート・ネーム

キーワードの入力時に、ショートネームが使えます。これは使用頻度が高く、かつ長い名前を選んで決められています。これらもキーワードになります。

ショートネームの表示 (LIST、GLIST等) は、CONTROLコマンドで指定します。(エディタの [BASIC mode] にて設定)

```
CONTROL 3 ; 0    --- フルネーム表示
           ; 1    --- ショートネーム表示
```

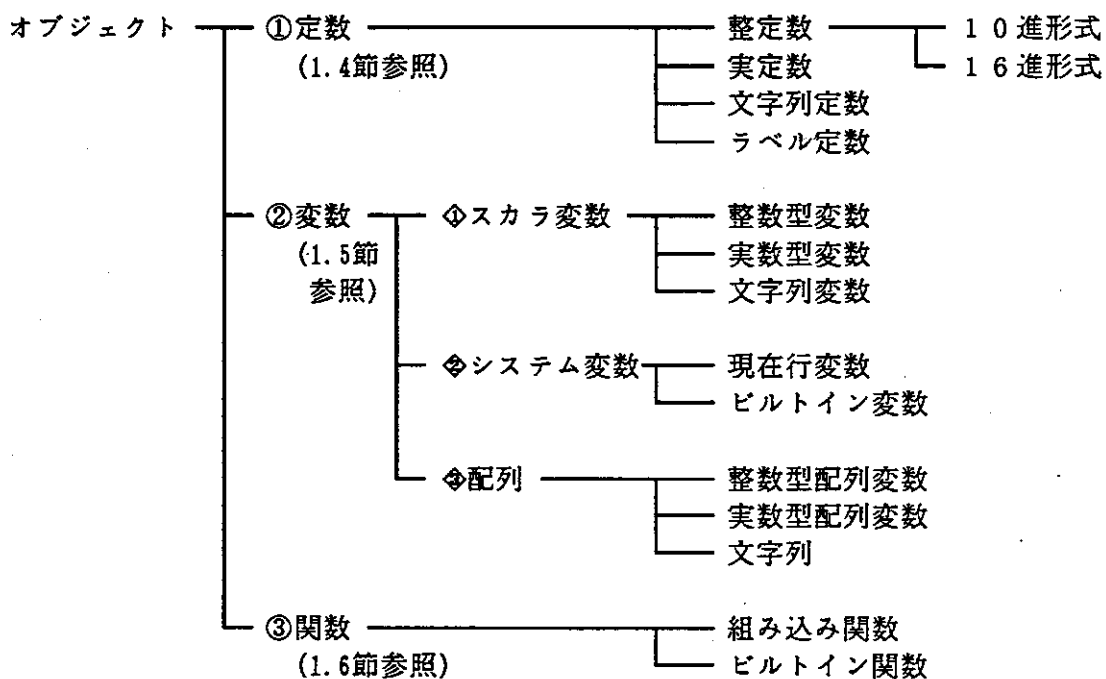
【フルネームとショートネームの対応表】

フルネーム	ショートネーム
CURSOR	CSR
ENTER	ENT
INITIALIZE	INIT
INPUT	INP
OUTPUT	OUT
PRINTF	PRF
USING	USE

1.3 オブジェクト

BASICが処理する対象となるものをオブジェクトと言います。オブジェクトには変数、定数、および関数があり、それぞれにデータ型があります。

変数型には整数型、実数型、および文字列型があります。



1.4 定数

1.4.1 整数

プログラムで、小数点のない数値は整数とみなします。

整数では、10進、16進で表現できます。

内部では4バイトにて表現するので、BASICでは10進数で

`2, 147, 483, 648 ~ -2, 147, 483, 647`

までの数値を表現できます。

先頭に0xを付けることにより16進で表現できます。

10進表現

例)

```
A = 123
PRINT 456
```

16進表現

例)

```
A = 0x10AB
PRINT 0x0000FFFF
```

1.4.2 実数

小数点付きや1E+20のような浮動小数点表現の数値は実数とみなします。

内部では8バイト(IEEE)で表現するので、BASICでは

`約 1E+308 ~ 1E-308`

まで表現でき、15桁の精度を持ちます。

例)

```
A = 123.0
B = 1.23456789
C = 1.23E6 - 9.87654
```

注) 整数どうしの演算

`PRINT 3/2` の場合、整数どうしの演算とみなして計算するので、値は1.0になります。

実数で演算させたい場合は、`PRINT 3.0/2.0` のように両方、またはどちらか一方に小数点を付けて演算させます。

1.4.3 文字列定数

255文字以内の文字群の前後をダブル・クォーテーション (") で囲んだものを文字列定数と言います。

文字列は、" " の空文字列から最大255文字まで表現できます。

キーボードに割り当てられていないコードをプログラムで表現するために、(\) を使って表現できます。またASCIIコードの制御文字を表現するために以下のエスケープ・シーケンスが用意されています。

注) " \ " を使うときは8進数、または下記のエスケープ・シーケンスで指定します。

エスケープ

シーケンス	8進	10進	
\b	010	8	バック・スペース
\t	011	9	水平タブ
\n	012	10	ライン・フィード
\v	013	11	垂直タブ
\f	014	12	フォーム・フィード
\r	015	13	キャリッジ・リターン

例)

```
A$="ABCDE"  
CR$="\r"  
NL$="\n"  
B$=""  
PRINT "ABCD\014"  
PRINT "AB\007"  
PRINT "ABC\n"
```

1.4.4 ラベル定数

文番号に代わるもので、ラベル名の先頭に* (アスタリスク) をつけて宣言します。ラベル名で使用できる文字は変数と同じですが、ラベルは変数ではないので数値を代入できません。またラベルが指定できる位置は構文的に限られていて、[2.3 コマンドとステートメントの説明]にあるラベル行番号または分岐先と書かれています。

1.5 変数

変数名は、英文字を先頭とするアルファニューメリックで構成し、最大20文字です。

注) 変数名は、~~文字列変数と同じではなりません。~~

【アルファニューメリック】

A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q
R, S, T, U, V, W, X, Y, Z
a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q
r, s, t, u, v, w, x, y, z
1, 2, 3, 4, 5, 6, 7, 8, 9, 0
_ (アンダー・バー)

変数名の最後に \$ を付けると文字列変数になります。

変数名の最後に (数字) を付けると配列変数になります。

変数は特に INTEGER 宣言されていなければ実数型になります。

例)

VAL		: 実数型変数
STRG\$: 文字列型変数
ARRY1 (4)		: 実数型配列変数
INTEGER	code 2	: 整数型変数
INTEGER	wk (7)	: 整数型配列変数

1.5.1 スカラ変数

- ・ 整数型変数
- ・ 実数型変数
- ・ 文字列型変数

のうち数値型、の変数は BASIC プログラム起動時に 0 で初期化されます。

従って特定の値に初期化する変数の場合は、プログラム内で明示的に値を代入する必要があります。

各データ型の変数に代入できる値の大きさは、定数で紹介した値と同じです。

~~文字列変数には、配列がありません。~~

文字列変数には文字列定数と同様に長さの属性があり、その長さを宣言するには、DIM 文を使います。

例)

DIM STRG\$ [100] : 長さ100文字の変数宣言

~~宣言をしなくても18文字分が予め宣言されています。~~

1.5.2 システム変数

・現在行変数 @

現在実行しているプログラムの行番号を格納しています。
値を代入することはできません。

例)

```
LIST @           : 現在実行している行を表示します。
```

・ビルトイン変数

BASICの起動時に自動的に登録される変数で、固有の値で初期化されます。また任意の値を代入して値を変更できます。起動時の値に戻すには明示的にその値を代入するか、またはプログラムをMOVE and RUNで初期化する必要があります。

```
PI      : 3.141592.....  
EXP     : 2.718281.....
```

・エラー番号変数

BASICのエラー番号を保持しているシステム変数です。

BASICのプログラムの開始時に0に初期化され、エラーが発生すると、その値が代入されます。

例)

```
PRINT ERRN
```

エラー番号は内部で以下のような構造になっています。

```
エラークラス * 256 + エラーメッセージ番号
```

```
エラークラス : 1   ; データ入出力関係  
              2   ; データ演算処理関係  
              3   ; ビルトイン関数関係  
              4   ; BASIC構文関係
```

詳しくは、1.6.1 組み込み関数のERRMSを参照して下さい。

1.5.3 配 列

配列の宣言には、DIM, INTEGER文を使います。

配列変数を宣言する際、複数の添字をコンマで区切って指定すると、その配列変数は添字の個数に応じた次元数をもつことになります。

(次元数は、最大10次元またはメモリ容量が許す限り)

・数値型配列

宣言しないで参照すると、その配列の大きさ、つまり要素数は10になります。

以下のように宣言したものと同じになります。

添字は必ず1から始まります。

宣言時に指定した添字が最大値となります。

```
DIM          AB (10)
INTEGER      CD (10)
```

例)

```
DIM          RL (30)           : 実数型配列変数宣言
INTEGER      IT (10, 20)       : 整数型配列変数宣言
                                   (2次元)
```

・文字列

文字列変数に格納される文字数をDIM文で宣言できます。

例)

```
DIM  A$ [100]           : 100文字まで代入可能
DIM  B$ [50]            : 50文字まで代入可能
```

注) 文字列変数の2次元配列はできません。

1.6 関数

1.6.1 組み込み関数

ERRM\$ (エラー番号)

パラメータで指定されたエラーメッセージを返します。
パラメータを0で指定すると、直前に表示されたエラーメッセージを返します。

エラー番号は内部で以下のような構造になっています。
エラークラス * 256 + エラーメッセージ番号

しかし、エラークラスを含む番号を指定しても、内部ではエラーメッセージ番号だけを参照します。したがって、エラー番号にERRNを指定できます。

例)

```
PRINT ERRM$ (ERRN)
: 直前のエラーメッセージを
出力します。
```

NUM (文字列式)

文字列式の先頭の1文字のASCIIコードを返します。

例)

```
NUM (" ABC ") → 65
A$ = " XYZ "
NUM (A$) → 88
```

CHR\$ (算術式)

算術式の値に対応するASCII文字1文字の文字列式を返します。

例)

```
CHR$ (65) → " A "
A = 88
CHR$ (A) → " X "
```

LEN (文字列式)

文字列式の長さを返します。

例)

```
LEN (" ADVANTEST ") → 9
A$ = " CORP. "
LEN (A$) → 5
```

POS (文字列式1、文字列式2)

文字列式1の中から、文字列式2がある位置の先頭の位置を返します。

例)

```
A$ = " AN "
POS (" ADVANTEST", A$) → 4
```

ABS (算術式) : 算術式の値の絶対値を返します。

例)

ABS (-1.2) → 1.2

ATN (算術式) : 算術式の値に対する逆正接値を返します。
(算術式はラジアン)

例)

ATN (PI) → 1.26262...

COS (算術式) : 算術式の値に対する余弦値を返します。
(算術式はラジアン)

例)

COS (PI) → -1.0

FRE (算術式) : BASICのメモリの残り容量(バイト)を返します。

例)

FRE (0) → 301458

(電源ON時: PRINT FRE (0)
をBASIC modeにて実行)

LOG (算術式) : 算術式の値に対する自然対数(eを底とした対数)を返します。

例)

LOG (EXP) → 1.0

SIN (算術式) : 算術式の値に対する正弦値を返します。
(算術式はラジアン)

例)

SIN (PI) → 0.0

SPOLL (算術式) : GPIB装置へのシリアル・ポールを行ない、ステータス・バイト値を返します。(算術式は0~31)

例)

SPOLL (2) : 外部接続GPIB装置のアドレス2のシリアル・ポールを行なう。

SPOLL (31) : 本器の計測部へのシリアル・ポールを行なう。

SQR (算術式) : 算術式の値に対する平方根を返します。

例)

SQR (2) → 1.41421356...

TAN (算術式) : 算術式の値に対する正接値を返します。
(算術式はラジアン)

例)

TAN (PI) → 1.0

項 目	ビルトイン関数
(1)周波数を求める	FREQ (P) DFREQ (P1, P2) FMAX (P1, P2, M) FMIN (P1, P2, M) BND (P, X, M) BNDL (P, X, M) BNDH (P, X, M) CBND (F, X, M) CBNDL (F, X, M) CBNDH (F, X, M) FRPLHN (N, M) FRPLLN (N, M)
(2)ポイントを求める	POINT (F) DPOINT (F1, F2) PMAX (P1, P2, M) PMIN (P1, P2, M) PRPLHN (N, M) PRPLLN (N, M)
(3)トレース・データを求める	LVPOINT (L) LVDPOINT (L1, L2)
(4)レベルを求める	VALUE (P, M) DVALUE (P1, P2, M) CVALUE (F1, F2, M) DCVALUE (F1, F2, M) LEVEL (T) DLEVEL (T1, T2) MAX (P1, P2, M) MIN (P1, P2, M) RPL1 (P1, P2, Dx, Dy, M) VRPLHN (N, M) VRPLLN (N, M)
(5)総電力を求める	POWER (P1, P2, M)
(6)リップル (極大、極小) の数を求める	NRPLH (P1, P2, Dx, Dy, M) NRPLL (P1, P2, Dx, Dy, M)
(7)上下限チェックを行なう	LMTMD1 (Dd, S, Ds) LMTMD2 (P, S, Ds, M) LMTUL1 (Dd, Up, Lo) LMTUL2 (P, Up, Lo, M)

項 目	ビルトイン関数
(8)トレース・データの入出力	RTRACE (P, M) WTRACE (T, P, M)
(9)グラフィック関係	GADRS (P, X, Y) GFLRECT (W, X1, Y1, X2, Y2) GLINE (L, W, X1, Y1, X2, Y2) GMKR (M, W, X, Y) GPOINT (W, X, Y) GRECT (L, W, X1, Y1, X2, Y2) GSTR (W, X, Y, 文字列式)

1.7 演算式

オブジェクトを操作するのが演算子です。演算子とオブジェクトの組合せで式を構成します。

演算子	(1) 代入演算子	(1.7.1項参照)
	(2) 単項算術演算子	(1.7.2項参照)
	(3) 2項算術演算子	(1.7.3項参照)
	(4) 論理演算子	(1.7.4項参照)
	(5) 比較演算子	(1.7.5項参照)
	(6) サブ・ストリング演算子	(1.7.6項参照)
	(7) ビット演算子	(1.7.7項参照)

1.7.1 代入演算子

従来のBASICにある LET文は、このBASICにはないので直接代入文が書けます。代入式は式自体に値を持ちます。

例)

```
A = 1 2 3
PRINT A
PRINT B$ = " ADVANTEST"
PRINT (A = 2) + A
```

【実行結果】

```
1 2 3. 0
ADVANTEST
4. 0
```

代入演算子を以下に示します。

代入演算子	使用例	意味
=	A = 1 2 3	普通の代入
+=	A += 5	A = A + 5 と同じ意味
-=	A -= 5	A = A - 5 と同じ意味
*=	A *= 5	A = A * 5 と同じ意味
/=	A /= 5	A = A / 5 と同じ意味
%=	A %= 5	A = A % 5 と同じ意味
=>	A \$=>" ABCD"	文字列を右詰めで代入
=<	A \$=<" ABCD"	文字列を左詰めで代入

注) % はモジュロ (余り)

例)

```
DIM S$ [15]
A = 5      :PRINT A
A += 10    :PRINT A
A -= 3     :PRINT A
A *= A     :PRINT A
A /= 2     :PRINT A
A %= 5     :PRINT A
PRINT " 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5"
S$ => " TEST" :PRINT S$
S$ =< " TEST" :PRINT S$
```

【実行結果】

```
5. 0
15. 0
12. 0
144. 0
72. 0
2. 0
1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
                        TEST
TEST
```

1.7.2 単項算術演算子

-	マイナス符号
+	プラス符号
++	<p>前置／後置 インクリメント (変数に1を加える)</p> <ul style="list-style-type: none"> ・前置インクリメント 例) $A = 2$ Aに2を代入 $B = ++A$ Aに1を加えてからBに代入 ・後置インクリメント 例) $A = 2$ Aに2を代入 $B = A++$ BにAを代入してからAに1を加える
--	<p>前置／後置 デクリメント (変数から1を引く)</p> <ul style="list-style-type: none"> ・前置デクリメント 例) $A = 2$ Aに2を代入 $B = --A$ Aから1を引いてからBに代入 ・後置デクリメント 例) $A = 2$ Aに2を代入 $B = A--$ BにAを代入してからAから1を引く

インクリメント演算子(++)を変数の前に置いたときは、変数の値が使われる前に+1の加算が行なわれます。変数の後に置いたときは、変数の値が使われてから+1の加算が行なわれます。デクリメント演算子(--)の場合も変数の値が1ずつ減算となる以外は同じです。

例)

```
A = 10
A++           : A = A + 1   同じ
++A          : A = A + 1   同じ
A--          : A = A - 1   同じ
--A          : A = A - 1   同じ
PRINT " 1", A++ : Aの値を表示してからAに1加える
PRINT " 2", A
B = --A      : Aから1引いてからBに代入
C = A++     : AをCに代入してからAに1加える
PRINT " 3", A
PRINT " 4", B
PRINT " 5", C
B = A--     : AをBに代入してからAから1を引く
PRINT " 6", A
PRINT " 7", B
C = -123
D = C + (-23) - (+50)
PRINT " 8", C
PRINT " 9", D
```

【実行結果】

```
1          10.0
2          11.0
3          11.0
4          10.0
5          10.0
6          10.0
7          11.0
8          -123.0
9          -196.0
```

1.7.3 2項算術演算子

+	加算
-	減算
*	積算
/	割り算
%	モジュロ (余り)
^	累乗
&	文字列の連結

例)

```
PRINT 10+2
A=10
PRINT A-5
PRINT A*A
PRINT 20/A
PRINT A%3
B=3^4
PRINT B
S$="ABCD"
S1$=S$&"EFG"
PRINT S1$
```

【実行結果】

```
12
5.0
100.0
2.0
1
81.0
ABCDEFG
```

1.7.4 論理演算子

複数の比較演算子を結合して複合条件を判定したりするのに使います。

NOT	否定	X		NOT X
		0		1
		1		0
AND	論理積	X	Y	X AND Y
		0	0	0
		0	1	0
		1	0	0
		1	1	1
OR	論理和	X	Y	X OR Y
		0	0	0
		0	1	1
		1	0	1
		1	1	1
XOR	排他的論理和	X	Y	X XOR Y
		0	0	0
		0	1	1
		1	0	1
		1	1	0

例)

```
IF NOT A THEN GOTO *MI
変数Aの値が0ならば、*MIへジャンプします。
```

```
IF X<100 OR 199<X THEN GOTO *LA
変数Xの値が100より小さい、または199より大きければ、*LAへ
ジャンプします。
```

```
IF 0<=X AND X<=100 THEN PRINT X
変数Xの値が0以上、かつ100以下のとき、変数Xの値をプリントしま
す。
```

```
IF A XOR B THEN PRINT A, B
変数Aの値が真(1)のとき変数Bの値が偽(0)、変数Aの値が偽(0)のとき変
数Bの値が真(1)であれば、変数A, 変数Bの値をプリントします。
```

1.7.5 比較演算子

比較演算子は2つの数値を比較するときに使います。比較結果は、真(1)偽(0)のどちらかで得られ、条件判断文(IF文)などのプログラムの流れを分岐させたりするのに使います。

IF文内の条件では、必ず論理演算を行なうので、“=”の演算子は無条件に比較演算子とみなします。したがって、代入式をIF文の条件式内に含めることはできません。

IF文の条件式以外で比較演算を行なうには、代入演算子の“=”と区別するために“==”をイコールとして使います。

例)

```
A = (B$=="ADVAN")
```

文字変数B\$が"ADVAN"ならば、変数Aには1が代入される。

記号	意味	例
= (または ==)	等しい	X=Y, X==Y
<>	等しくない	X<>Y
<	小さい	X<Y
>	大きい	X>Y
<=	小さいか等しい	X<=Y
>=	大きいか等しい	X>=Y

注) <=、>=の記号は<、>にはなりません。

例)

```
A = 1
```

```
B = 2
```

```
IF A=1 AND B>1 THEN PRINT "A"
```

```
IF A<>1 OR B=5 THEN PRINT "B"
```

```
IF NOT A THEN PRINT "C"
```

```
IF A XOR B>=3 THEN PRINT "D"
```

```
IF A==(B-1) THEN PRINT "E"
```

【実行結果】

```
A
```

```
D
```

```
E
```

1.7.6 サブ・ストリング演算子

文字列式の1部を参照できます。

・文字列式 [算術式1, 算術式2]

文字列式の先頭より算術式1の値だけ進んだところから、算術式2の値のところまでをサブ・ストリングとします。

例)

```
A$="ABCDEFGG"  
PRINT A$ [3, 5]  
PRINT "*ADVANTEST*" [2, 6]  
PRINT "*ADVANTEST*" [7, 10]
```

【実行結果】

```
CDE  
ADVAN  
TEST
```

・文字列式 [算術式1 ; 算術式2]

文字列式の先頭より算術式1の値だけ進んだところから、算術式2の値の文字数分までをサブ・ストリングとします。

例)

```
A$="ABCDEFGG"  
PRINT A$ [3 ; 4]  
PRINT "*ADVANTEST*" [7 ; 4]  
PRINT "*ADVANTEST*" [2 ; 5]
```

【実行結果】

```
CDEF  
TEST  
ADVAN
```


1.7.7 ビット演算子

ビット演算子はデータを構成する各ビットに対して、直接働く演算子です。ビット演算子は各ビットごとに論理演算 (AND, OR, XORなど) を行ないます。ビット演算は、必ず整数型にて行ないます。ビット演算ができる範囲は0~65535の16ビット内です。負数を指定した場合、エラーになります。

(NO operand in ...)

数値変数で使用するときは、2.3節(24)のINTEGER命令で整数型を宣言して下さい。

BNOT	(1の補数) BNOT 0 → 65535 BNOT 65535 → 0
BAND	(論理積) 65535 BAND 255 → 255 255 BAND 1024 → 0
BOR	(論理和) 255 BOR 1024 → 1279 1 BOR 2 → 3
BXOR	(排他的論理和) 255 BXOR 128 → 127 1 BXOR 3 → 2

例)

```

INTEGER S
*L
S=SPOLL(31)
IF S BAND 4 THEN PRINT "SWEEP END"
GOTO *L

```

1.8 演算の優先順位

種々の演算には優先順位があります。
実行時には下表の番号順に処理されます。

優先順位	演算子	優先順位	演算子
1	() で囲まれた式	10	BNOT
2	関数	11	BAND
3	~	12	BOR
4	符号	13	BXOR
5	++, --	14	NOT
6	*, /	15	AND
7	%	16	OR
8	+, -	17	XOR
9	関係演算子		

1.9 文字列演算

BASICでは、文字列に対して演算ができます。

1.9.1 文字列の連結

文字列は、" & " で連結できます。

```
例)   A$=" ADVAN" &" TEST"  
       B$=A$&"  co. "  
       PRINT A$  
       PRINT B$
```

```
実行結果)  ADVANTEST  
            ADVANTEST  co.
```

1.9.2 文字列の比較

文字も、数値の比較と同様に、関係演算子を用いて比較できます。

=, <, >, <>, <=, >=

IF文以外で、=を比較するときは ==を使用します。

```
例)   PRINT A==B
```

文字列の最初から1文字ずつ、比較します。相互に同じ長さの文字列の場合は、その文字のASCIIコードが大きい方の文字列が大きいと判断します。文字列の片方が短い場合は、短い方の文字列を小さいと判断します。

文字列の場合は、スペースなども意味を持つので注意して下さい。

```
例) " AA" =" AA"           → 真 (ture)  
    " AA" < " aa"         → 真 (ture)  
    " AAA" > " AA"        → 真 (ture)
```

1.9.3 型変換

文字列表現式から数値変数、数値表現式から文字列変数への代入は、ダイレクトに行なえます。

文字列表現式から数値変数の場合、文字列の最初に数字があったところから数字がなくなったところまでとします。

```
例)   A=" 1 2 3. 4"  
       B=" ABC4 5 6. 7 DEF 8 9"  
       C$=1 2 3  
       D$=B  
       PRINT A  
       PRINT B  
       PRINT C$  
       PRINT D$
```

実行結果)

1 2 3 . 4

4 5 6 . 7

1 2 3

4 5 6 . 7

2. コマンドとステートメントの文法と解説

2.1 概要

この章では、本器で使われるコマンドやステートメントの構文を直観的に理解できるように、記述式表現で解説します。

2.2 はじめに

2.2.1 説明の構成

各命令の解説は次の構成で説明します。

B U Z Z E R	--	命令の名前 (ショート・ネーム:省略形)
【概要】	--	命令の機能説明
【書式】	--	命令の記述の仕方 (記述表現式)
【解説】	--	命令の使用法や詳しい機能の説明
【例】	--	命令の文例
【注意】	--	注意書き
【プログラム例】	--	その命令を使用したプログラム例
【実行結果】	--	プログラム例の実行結果

(1) 書式

【書式】の記述式表現には、以下に示す記号を使用しています。

- < > : この記号で囲まれた部分は、ユーザが指定します。
- [] : この記号で囲まれた部分は、省略できます。
- { } : この記号で囲まれた部分は、繰り返し用いることができます。
- , : コンマで複数の指定ができます。
- | : または の意味

例)

<A> | . . . <A>またはを用いる、ということ
ことです。

(2) 解説

【解説】に使用している単語の意味を説明します。

数値表現式 : 数値定数、数値変数、数式のいずれかを示します。

文字列表現式 : 文字列定数、文字列変数、文字列関数、サブ・ストリングで構成されている文字列式

装置アドレス : GPIBに接続されている装置のGPIBアドレス

ファイル・ディスクリプタ

: 変数と同等の扱いで、データの入出力文につけます。

ラベル : ラベル名 (行番号も含む)

ラベル名は、アスタリスク (*) を先頭につけ、英文字で構成します。

(3) 機能別コマンドとステートメント

(1/3)

機 能	コマンドとステートメント	内 容
コ マ ン ド	CONT CONTROL LIST LLIST LISTN LLISTN RUN SCRATCH STEP	プログラム停止後の再実行を行なう。 各制御に関する値を設定する。 プログラム・リストを表示する。 プログラム・リストを表示する。(RS232C) プログラム・リストを表示する。 プログラム・リストを表示する。(RS232C) プログラムを実行させる。 BASIC内のプログラムを消去する。 プログラムを1行実行する。
算 術 関 数	ABS ATN COS LOG SIN SQR TAN	与えられた値の絶対値を求める。 与えられた値の逆正接値を求める。 与えられた値の余弦値を求める。 与えられた値の自然対数を求める。 与えられた値の正弦値を求める。 与えられた値の平方根を求める。 与えられた値の正接値を求める。
ビット演算	BAND BNOT BOR BXOR	ビットANDを求める。 ビットNOTを求める。 ビットORを求める。 ビットXORを求める。
割り込み制御	ENABLE INTR DISABLE INTR ON END ON KEY ON ISRQ ON SRQ ON ERROR OFF END OFF KEY OFF ISRQ OFF SRQ OFF ERROR	割り込み受信許可状態にする。 割り込み受信禁止状態にする。 EOF割り込みの分岐を定義する。 キー割り込みの分岐を定義する。 本体計測部SRQ割り込みの分岐を定義する。 GPIB、SRQ割り込みの分岐を定義する。 エラー発生割り込みの分岐を定義する。 EOF割り込みの分岐定義を解除する。 キー割り込みの分岐定義を解除する。 本体計測部SRQ割り込みの分岐定義を解除する。 GPIB、SRQ割り込みの分岐定義を解除する。 エラー発生割り込みの分岐定義を解除する。
文字列操作	NUM CHRS LEN POS SPRINTF	文字列の先頭の文字のASCIIコードを求める。 数値をASCII文字に変換する。 文字列の長さを求める。 文字列2の中から文字列1がある位置を求める。 書式を決め文字列変数に代入する。

注) 各コマンドとステートメントの説明は2.3節を参照して下さい。

機能	コマンドとステートメント	内容
メモリカード 制御	CAT CALL CLOSE # ENTER # OPEN # OUTPUT # INITIALIZE(INIT) PURGE RENAME	メモリ・カードの内容を表示する。 サブ・プログラムをロードする。 ファイルを閉じる。 ファイルからデータを読み込む。 ファイルを開く。 ファイルにデータを書き込む。 メモリ・カードを初期化する。 指定ファイルを消去する。 ファイル名を変更する。
画面制御	CURSOR(CSR) CLS	指定位置にカーソルを移動する。 画面を消去する。
ステートメント	BUZZER DIM FOR TO STEP NEXT BREAK CONTINUE GOSUB RETURN GOTO IF THEN ELSE END IF INPUT(INP) INTEGER LPRINT LPRINT USING(USE) PAUSE PRINT(?) PRINT USING(USE) PRINTER PRINTF(PRF) READ DATA RESTORE REM(!) SELECT CASE END SELECT STOP WAIT	ブザーを鳴らす。 配列変数を宣言する。 繰り返し処理を設定する。 繰り返し処理から抜ける。 繰り返し処理を先頭に戻す。 サブ・ルーチン分岐する。 サブ・ルーチンからの復帰する。 指定位置へ分岐する。 条件判断をして処理する。 変数への入力を行なう。 整数型変数の宣言する。 プリンタ(RS-232)出力する。 プリンタ(RS-232)出力する。(書式指定) 一時停止する。 文字を画面に表示する。 文字を画面に表示する。(書式指定) GPIBプリンタ装置のアドレス指定を行なう。 文字を画面に表示する。(書式指定) DATA文からデータを読み取り、変数へ代入する。 READ文で読むDATA文を指定する。 注釈文 条件判断を行い処理する。 プログラム実行停止する。 指定時間、実行停止する。

注) 各コマンドとステートメントの説明は2.3節を参照して下さい。

機 能	コマンドとステートメント	内 容
G P I B関係 コ マ ン ド	CLEAR DELIMITER ENTER(ENT) GLIST GLISTN GPRINT GPRINT USING(USE) INTERFACE CLEAR LOCAL LOCAL LOCKOUT OUTPUT(OUT) REMOTE REQUEST SEND SPOLL TRIGGER	D C L、S D Cを送出する。 デリミタを設定する。 G P I Bデータを入力する。(パラレルI/O含む) G P I Bプリンタへプログラム・リストを出力する。 G P I Bプリンタへプログラム・リストを出力する。 データをG P I Bプリンタへ出力する。 データをG P I Bプリンタへ出力する。 (書式指定) I F Cを送出する。 指定装置をローカル状態にする。 指定装置をローカル・ロックアウト状態にする。 データをG P I Bに出力する。 (パラレルI/O含む) 指定装置をリモート状態にする。 標準G P I BにS R Qを出力する。 G P I Bデータを1つずつ出力する。 指定装置のシリアル・ポールを行なう。 G E Tを送出する。

注) 各コマンドとステートメントの説明は2.3節を参照して下さい。

2.3 各種コマンドとステートメントの説明

以下の順（アルファベット順）に説明します。

- | | |
|---|--|
| (1) BUZZER | |
| (2) CALL | |
| (3) CAT | |
| (4) CLEAR | |
| (5) CLS | |
| (6) CLOSE # | |
| (7) CONT | |
| (8) CONTROL | |
| (9) CVRSOR (CSRと省略可) | |
| (10) DELIMITER | |
| (11) DIM | |
| (12) DISABLE INTR | |
| (13) ENABLE INTR | |
| (14) ENTER # | |
| (15) ENTER (ENTと省略可) | |
| (16) FOR...TO...STEP
BREAK/CONTINUE~NEXT | |
| (17) GLIST | |
| (18) LISTN | |
| (19) GOSUB~RETURN | |
| (20) GOTO | |
| (21) GPRINT | |
| (22) GPRINT USING (USEと省略可) | |
| (23) IF THEN ELSE/
ELSE IF/END IF | |
| (24) INITIALIZE (INITと省略可) | |
| (25) INTEGER | |
| (26) INTERFACE CLEAR | |
| (27) INPUT (INPと省略可) | |
| (28) LIST | |
| (29) LISTN | |
| (30) LLIST | |
| (31) LLISTN | |
| (32) LOCAL | |
| (33) LOCAL LOCKOUT | |
| (34) LPRINT | |
| (35) LPRINT USING (USEと省略可) | |
| (36) OFF END | |
| (37) OFF ERROR | |
| (38) OFF KEY | |
| (39) OFF SRQ/ISRQ | |
| (40) ON END ~
GOTO/GOSUB | |
| (41) ON ERROR ~
GOTO/GOSUB | |
| (42) ON KEY ~
GOTO/GOSUB | |
| (43) ON SRQ/ISRQ ~
GOTO/GOSUB | |
| (44) OPEN # | |
| (45) OUTPUT # | |
| (46) OUTPUT (OUTと省略可) | |
| (47) PAUSE | |
| (48) PPRINTF (PRFと省略可) | |
| (49) PRINT (?と省略可) | |
| (50) PRINT USING (USEと省略可) | |
| (51) PRINTER | |
| (52) PURGE | |
| (53) READ DATA/RESTORE | |
| (54) REM (!と省略可) | |
| (55) REMOTE | |
| (56) RENAME | |
| (57) REQUEST | |
| (58) RUN | |
| (59) SCRATCH | |
| (60) SELECT CASE/
CASE ELSE/END SELECT | |
| (61) SEND | |
| (62) SPOLL | |
| (63) SPRINTF | |
| (64) STEP | |
| (65) STOP | |
| (66) TRIGGER | |
| (67) WAIT | |

(1) BUZZER

【概要】

本器内臓のブザーを鳴らします。

【書式】

BUZZER <音程>, <時間>

【解説】

指定された<音程>を指定された<時間>でならします。

音程 : 整数 0~65535 (Hz)
時間 : 整数 0~65535 (msec)

【音程 (目安の数値)】

261	: ド	C	277	: ド#	C#
294	: レ	D	311	: レ#	D#
330	: ミ	E			
349	: ファ	F	370	: ファ#	F#
392	: ソ	G	415	: ソ#	G#
440	: ラ	A	466	: ラ#	A#
494	: シ	B			

数値を2倍にすると1オクターブ上がり、 $\frac{1}{2}$ 倍にすると1オクターブ下がります。数値を大きくすると高音になります。

【例】

BUZZER 440, 1000 : ラの音を1秒鳴らす。

【注意】

BUZZER命令は、ブザーをならす回路に設定するだけなので、ブザーは、指定時間鳴りますが、この命令の実行はすぐに終了します。

(1秒間で設定してもBUZZER命令は、設定後すぐに次の命令を実行します。)

連続してブザーをならすときは、WAIT命令をBUZZER命令の次にBUZZER命令を同じ時間だけ設定して下さい。

【プログラム例】

```
FOR I=1 TO 8  
  READ S  
  BUZZER S, 1000  
  WAIT 1000  
NEXT I  
DATA 261, 294, 330, 349, 392, 440  
DATA 494, 523
```

【実行結果】

”ドレミファソラシド”を1秒ごとに音階を変えて鳴らします。

(2) CALL

【概要】

作成済みのプログラムの最終に、メモリ・カードに登録されたファイル（サブプログラム）を読み込み、追加し、サブルーチンとして実行します。ファイル名は、最大10文字までです。

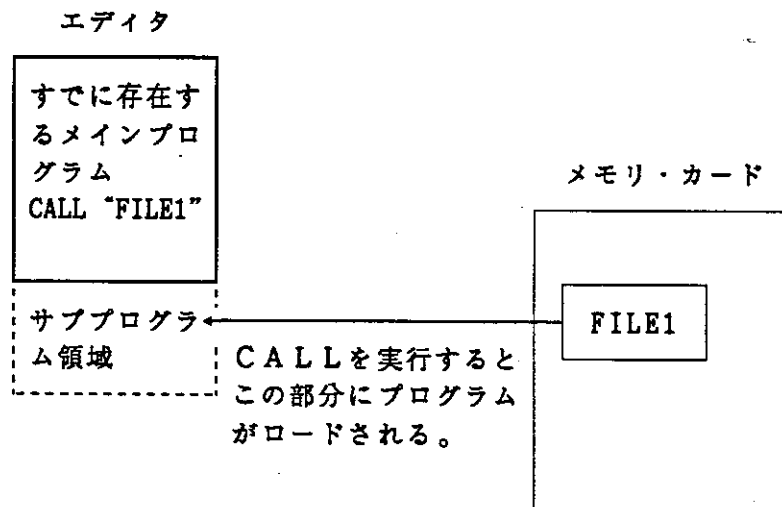
【書式】

CALL <ファイル名>

【解説】

すでに本器に存在するプログラムをメインプログラムとし、このメインプログラムの中のCALL命令でメモリ・カードから読み込み、メインプログラムの最終にプログラムを追加します。その追加されたプログラムをサブプログラムとします。（サブプログラムはサブルーチンとして扱います。）

メインプログラムからCALL命令でサブプログラムを読み込み、サブルーチンとして実行した後にRETURN（復帰）し、CALL命令の次の文から実行を再開します。（（19）GOSUB～RETURNと同様）



【例】

CALL "FILE1" : FILE1というプログラムを読み込み実行する。

【注意】

サブプログラム中にはCALL命令は使用できません。使用した場合にはエラーとなり、実行を中止します。

(CANNOT nest CALL)

サブプログラムは実行を終了するとロードしたサブプログラムを削除してメインプログラムにRETURN（復帰）します。従ってエディタ上ではメインプログラム以外に何も表示されません。

変数は、メインプログラムと共用になるため、FOR等のループ処理での同じ変数名は誤動作の原因となります。サブプログラム専用の変数（ローカル変数）は作成できません。

サブプログラムにする場合は、サブプログラムをロードする際にRETURN命令を自動的に挿入します。RETRUNがすでに存在していても動作には影響はありません。

サブプログラムに行番号がある場合は、ロードする際に行番号を無視します。メインプログラムと行番号が重複していても動作します。

【プログラム例】

```
FOR I=1 TO 10      : このプログラムを
  PRINT I;         : "FILE1"として
NEXT I             : SAVEする。
```

```
FOR K=1 TO 5      : このプログラムを
  CALL "FILE1"    : メインプログラムとして
  PRINT           : RUNさせる。
NEXT K            :
```

【実行結果】

```
loading...
12345678910
loading...
12345678910
loading...
12345678910
loading...
12345678910
loading...
12345678910
Program ended normally.
```

(3) C A T

【概要】

メモリ・カードに格納されているファイル名を出力します。

【書式】

C A T

【解説】

- メモリ・カードに格納されているファイル名を出力します。
- エディタの **BASIC mode** で C A T 命令を実行します。
- C A T を実行すると以下の通り、管面に出力されます。

```
Card:032k      Page:02/02      :現在のメモリ・サイズ／ページ
File:023/024  MAXbyte:025600  :登録ファイル数／最大メモリ数
NO.+ File-Name + Type      + Byte
013+ T          + P          + 000051
014+ RIPPLE_CHK+ P          + 000405
015+ PLOT       + D          + 001802
-----
```

----- 各ファイルのサイズ (バイト)
----- ファイル・タイプ
P: BASICプログラム
D: BASICデータ
1~8: 設定データ関係
----- ファイル名(10文字)
----- 番号
----- (省略)

```
023+ DF_DP_CHK + P          + 000766
024+ DATA     + 1 3        + 000436 ----- 設定データ(1)、トレース
                                         ・データ(3)を示します。

Page= 1:next / 2:prev / 3:end
```

----- 操作 1: 次ページへ進む
2: 前ページへ戻る
3: 終了

- ・ 設定データ関係を以下に示します。
 - 1 : 設定データ
 - 2 : メニュー (ユーザ・デファイン)
 - 3 : トレース・データ A
 - 4 : トレース・データ B
 - 5 : ノーマライズ・データ A
 - 6 : ノーマライズ・データ B
 - 7 : リミット・ライン・データ
 - 8 : アンテナ・データ

【例】

CAT

【注意】

- ・ 新しいメモリ・カードは (23) の INITIALIZE 命令で初期化しないと使えません。

(4) CLEAR

【概要】

GPIB上に接続されたすべての装置、または指定した特定の装置を初期状態にします。

【書式】

CLEAR [装置アドレス {, 装置アドレス}]

【解説】

- ・ 装置アドレスを指定せずにCLEAR文だけ実行すると、GPIB上にユニバーサル・コマンドのデバイス・クリア(DCL)を送ります。このコマンドでGPIB上のすべての装置を初期状態にできます。
- ・ CLEAR文に続いて装置アドレス(0~30)を指定すると、装置アドレスによって指定されている装置のみに、ユニバーサル・コマンドのセレクトデバイス・クリア(SDC)を送ります。これにより特定の装置のみを初期状態にできます。なお、装置アドレスは複数指定できます。

【例】

CLEAR

CLEAR 1

CLEAR 2, 5, 8

【注意】

- ・ 装置アドレスで0~30以外を指定すると、エラーになり、プログラムの実行を中止します。
(UNIT addr error in CLEAR)

【プログラム例】

【実行結果】

CLEAR 3	: 装置アドレス3が初期状態になる
OUTPUT 3;"CF3MZ"	: 装置アドレス3にデータを送信する
CLEAR 3,5	: 装置アドレス3と5が初期状態になる
OUTPUT 3,5;"CF2MZ"	: 装置アドレス3と5にデータを送信する
CLEAR	: 接続されているすべての装置が初期状態になる

(5) C L S

【概要】

外部端末、本体画面をクリアします。

【書式】

C L S [1 | 2]

【解説】

- ・ 外部端末、本体画面のキャラクタ画面、グラフィック画面をクリアします。
- ・ C L S 命令のパラメータは、以下のようになります。

指定なし : キャラクタ画面のみ
1 : グラフィック画面のみ
2 : キャラクタ、グラフィックの両画面

【例】

C L S

C L S 1

C L S 2

【注意】

- ・ グラフィック画面を使う前には、必ずOUTPUT 31命令((46)OUTPUT参照)でGP I Bコードの" V S 3" (3.2.1項(2)グラフィック機能参照)を送って下さい。

【プログラム例】

その1

```
OUTPUT 31;"VS3"  
GLINE(0,1,0,0,1023,439)  
PRINT "TEST"  
WAIT 2000  
CLS
```

その2

```
OUTPUT 31;"VS3"  
GLINE(0,1,0,0,1023,439)  
PRINT "TEST"  
WAIT 2000  
CLS 1
```

その3

```
OUTPUT 31;"VS3"  
GLINE(0.1,0,0,1023,439)  
PRINT "TEST"  
WAIT 2000  
CLS 2
```

【実行結果】

その1

文字列"TEST"が消去され、グラフィックが残ります。

その2

グラフィックが消去され、文字列"TEST"が残ります。

その3

グラフィック、文字列"TEST"がともに消去されます。

(6) CLOSE

【概要】

ファイル・ディスクリプタに割り当てられているファイルをクローズします。

【書式】

CLOSE #ファイル・ディスクリプタ

【解説】

- OPEN命令 (44) OPEN参照) でオープンしたファイルは、メモリ・カードを抜く前や、本器の電源をOFFする前に、必ずすべてのファイルをクローズさせなければ、ファイルが破壊されます。特に書き込みとしてオープンしたファイルは、必ずクローズして下さい。
- BASICプログラムでは、(47) PAUSE命令やSTOPキーでプログラムの実行を停止させたときは、ファイルを自動的にクローズしません。それ以外のときはプログラムの終了とともにすべてのファイルを自動的にクローズします。
エラー終了時もクローズしますが、ON ERRORの定義(41) ON ERROR参照) がある場合はクローズしません。
以上のような理由から、エラー終了時には以下の方法で明示的にクローズ動作を実行して下さい。

CLOSE *

このコマンドですべてのファイルをクローズします。

【例】

CLOSE #FD

CLOSE *

【注意】

- ファイルをオープンしたときには、最後にクローズを必ず実行して下さい。

【プログラム例】

```
OPEN "FFF" FOR OUTPUT AS #FD
FOR I=1 TO 100
  OUTPUT #FD;I
NEXT I
CLOSE #FD
OPEN "FFF" FOR INPUT AS #FD
FOR I=1 TO 100
  ENTER #FD;N
  PRINT N
NEXT I
CLOSE #FD
```

【実行結果】

1から100までの実数を"FFF"というファイル名で書き込み、終了したら読み出し、書き込んだデータを表示します。(1.0~100.0までプリントします。)

(7) CONT

【概要】

BASICプログラムの実行を再開させます。

【書式】

CONT [<ラベル>]

【解説】

- ・ ラベルを省略すると、プログラムの実行が停止した次の行から実行を再開します。ただしプログラムの実行が終了しているときは、エラーになります。
(Program CANNOT be continued)
- ・ ラベルを指定すると、指定ラベル位置の次の命令から実行を開始します。
- ・ ラベルの文字数は、最大20文字を目安にして下さい。

【例】

```
CONT
```

```
CONT *ABC
```

【注意】

- ・ ラベル指定のCONT命令の実行は、BASIC modeで行なって下さい。(その他の実行は、外部端末のF20キー、または本体CRTのCONTキーを押して下さい。)
- ・ CONT命令での実行は、変数の初期化をしません。
- ・ プログラムの実行を一時停止させるには、(46)のPAUSE命令を使います。

(8) CONTROL

【概要】

BASICの制御に関する値を設定します。

【書式】

CONTROL <レジスタ番号>; <数値表現式>

<レジスタ番号>: 1、2、3 のいずれかを指定をします。

【解説】

- レジスタ番号 1 : シリアルI/Oポート (RS-232) の設定をします。
2 : プリンタ出力時の左マージンを指定をします。
3 : プリンタ出力時にショート・ネームで出力するか否かを指定をします。

・レジスタ 1

シリアルI/Oポート (RS-232) の設定をします。
設定は以下の設定項目の値を合計し、
CONTROL 1; <X:合計した値>
で設定します。

ボーレート	1 : 2400 ボー 2 : 4800 ボー 3 (0) : 9600 ボー ◎
キャラクタ長	0 : 5 ビット 4 : 6 ビット 8 : 7 ビット 12 : 8 ビット ◎
パリティ・チェック	0 : パリティなし ◎ 16 : 奇数パリティ 32 : 偶数パリティ
ストップ・ビット	0 : なし 64 : 1 ビット ◎ 128 : 1.5 ビット 192 : 2 ビット

◎印: 工場出荷時の設定状態

・レジスタ 2

LISTの出力を指定数分だけスペースをあけ出力を右に寄せます。

・レジスタ 3

LIST出力をショート・ネームで出力するか、フル・ネームで出力するかを設定します。

0 : フル・ネーム出力
1 : ショート・ネーム出力

【例】

レジスタ 1

CONTROL 1 ; 0 + 1 2 + 0 + 6 4

または

CONTROL 1 ; 7 6

で設定すると、以下のように設定されます。

ボーレート	9 6 0 0 ボー
キャラクタ長	8 ビット
パリティ	なし
レジストップ・ビット	1 ビット

レジスタ 2

プリンタ出力でスペースを5個入れたいとき

CONTROL 2 ; 5

で設定します。

```
____10 PRINT "ADVANTEST"  
____20 FOR I=1 TO 10  
____30 PRINT I  
____40 NEXT I
```


レジスタ 3

ショート・ネームでプリンタに出力します。

CONTROL 3 ; 1

で設定します。

```
10 CSR 10,10
20 OUT 31:"CF1.23MZ"
30 ENT 31:A
40 PRF "%f",A
```

フル・ネームでプリンタに出力します。

CONTROL 3 ; 0

で設定します。

```
10 CUSOR 10,10
20 OUTPUT 31:"CF1.23MZ"
30 ENTER 31:A
40 PRINTF "%f",A
```

【注意】

- CONTROL 命令でシリアル I/O ポートの設定を変更しても、本体のパネルによる設定の表示は変更されません。なお、電源 ON 時には本体パネルの RS-232 設定項目のパラメータで設定されます。

(9) CURSOR (CSR と省略可)

【概要】

画面のカーソルを指定位置へ移動させます。

【書式】

CURSOR <X:カラム>, <Y:行>

CSR <X:カラム>, <Y:行>

【解説】

カラム、行で指定された位置へカーソルを移動します。

	端末あり	端末なし
<X:カラム> (整数)	$0 \leq X \leq 79$	$0 \leq X \leq 53$
<Y:行> (整数)	$0 \leq Y \leq 23$	$0 \leq Y \leq 23$

端末あり: 外部ターミナル接続状態での起動

端末なし: 計測器のみでの起動

どちらのCRTでも左上が(0, 0)になります。

53

79

(0,0)	本体CRT	外部 ターミナル
23	(23, 53)	(23, 79)

【例】

CURSOR 10, 20 : 10カラム、20行目にカーソルを移動

CSR 50, 11 : 50カラム、11行目にカーソルを移動

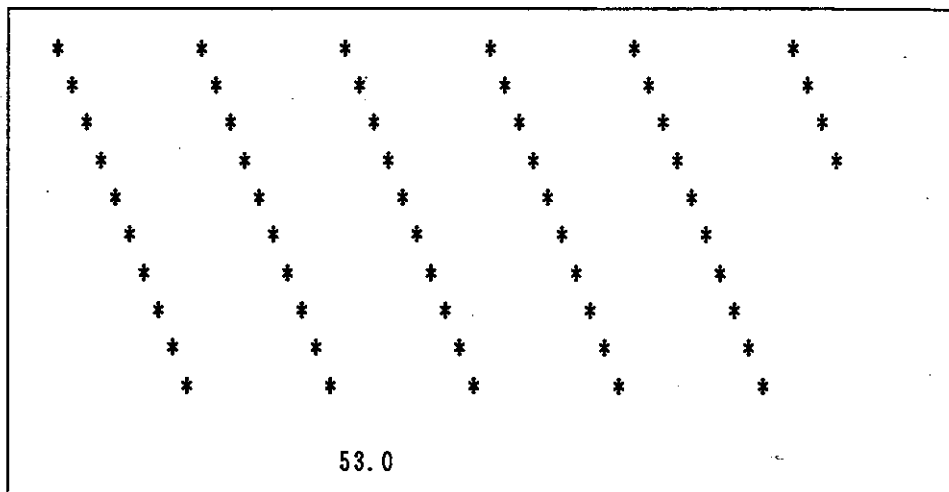
【注意】

端末なし(計測器CRT)の場合、BASICで使用できる画面サイズのカラム(X軸)が小さくなります。

【プログラム例】

```
OUTPUT 31;" VS2"  
FOR X=0 TO 53  
  CURSOR X, X%10  
  PRINT "*" ;  
  CSR 20,15:PRINT X  
NEXT X
```

【実行結果】



(10) DELIMITER

【概要】

4種類のデリミタを選択し、設定します。

【書式】

DELIMITER <X>

X: 数値表現式 0、1、2、3のいずれか

【解説】

- OUTPUT 0～30命令((46)OUTPUT参照)で出力時のターミネータ(デリミタ)を設定します。
- デリミタの選択番号および種類を下表に示します。

選択番号	デリミタの種類
0	"CR", "LF" + 単線信号EOIの2バイトを出力
1	"LF" の1バイトを出力
2	データの最終バイトと同時に単線信号EOIを出力
3 ◎	"CR", "LF" の2バイトを出力

(◎印は電源ON時のデフォルト値)

【例】

DELIMITER 0

DELIMITER 1

DELIMITER 2

DELIMITER 3

【注意】

- ・ 選択番号 0 ~ 3 以外を指定すると、エラーになり、プログラムの実行を中止します。
(invalid value in DELIMITER)
- ・ DELIMITER 文のデフォルト値は 3 です。

【プログラム例】

【実行結果】

DELIMITER 0	: デリミタを CR, LF + EOI に設定する
OUTPUT 3; "CF3MZ"	: 装置アドレス 3 にデータを出力
DELIMITER 1	: デリミタを LF に設定
OUTPUT 3; "CF4MZ"	: 装置アドレス 3 にデータを出力
DELIMITER 2	: デリミタを最終バイト + EOI に設定する
OUTPUT 3; "CF5MZ"	: 装置アドレス 3 にデータを出力
DELIMITER 3	: デリミタを CR, LF に設定する
OUTPUT 3; "CF6MZ"	: 装置アドレス 3 にデータを出力

(11) DIM

【概要】

配列変数または文字列変数の大きさを宣言します。

【書式】

DIM <X> {, <X>}

X : 変数名 (<数値表現式> {, <数値表現式>}) |
文字列変数名 ["<数値表現式>"]

【解説】

- 配列変数および文字列変数の大きさを宣言します。配列の要素を示す添字は最大値を指定します。(宣言をしないで使用すると、配列では1次元につき最大値10の要素数になり、文字列では18文字の長さで使用できます)
- DIM命令で配列宣言すると、指定された大きさの配列変数をメモリ上に確保します。したがって、大きすぎる配列宣言を行なうとメモリ領域が足りなくなり、エラーとしてプログラムの実行を中止します。
(memory space full)
- 配列変数のときの大きさを示す数値表現式は、実数表現になっていても、小数点以下を切り捨てて整数として扱い宣言、参照します。
- 文字列変数のときの大きさを示す数値表現式は文字列の長さを宣言します。
- 添字を複数個指定すると個数分の次元を持つ配列変数の指定となります。
(次元数は、メモリ容量が許す限り)

【例】

DIM A (20) : 実数型配列変数Aを20個確保する。
DIM B (30), C (25, 5) : 実数型配列変数Bを30個、実数型2次元配列変数Cを25*5個確保する。
DIM S\$ [50] : 文字列変数S\$を50文字分確保する。
DIM D (15), T\$ [80] : 実数型配列変数Dを15個、文字列変数T\$を50文字分確保する。
DIM E (3, 8, 5, 2) : 実数型3次元配列変数Eを3*5*2個確保する。

【注意】

- ・ 配列変数の添字の最小値は、1です。
- ・ 配列宣言の大きさは、プログラムの大きさにより確保できる量が変わります。
- ・ 文字列変数の長さは、最大128です。
- ・ 整数型変数を配列宣言するときは、(25) INTEGER命令を参照して下さい。
- ・ 添字が指定範囲外、0または負数で配列変数を参照すると、エラーになります。
(Array's range error または
Invalid dimension parameter)
- ・ 文字列変数における多次元配列宣言はできません。

【プログラム例】

```
DIM A(20),B(5,4),SS[20]           : 実数型配列宣言、文字列変数宣言
FOR I=1 TO 20                     : Iが1からI>20になるまでループ
  A(I)=I                           : 配列変数にIを代入
NEXT I                             : NEXT I
FOR X=1 TO 5                       : Xが1からX>5になるまでループ
  FOR Y=1 TO 4                     : Yが1からY>4になるまでループ
    B(X,Y)=A((Y-1)*5+X)           : 配列Bに配列Aを代入
  NEXT Y                           : NEXT Y
NEXT X                             : NEXT X
SS="ABCDEFGHIJKLMNOPQRSTUVWXYZ"   : S $に文字列26文字を代入
FOR X=1 TO 5                       : Xが1からX>5になるまでループ
  FOR Y=1 TO 4                     : Yが1からY>4になるまでループ
    CURSOR X*5,Y:PRINT B(X,Y)     : 配列Bを出力
  NEXT Y                           : NEXT Y
NEXT X                             : NEXT X
PRINT SS                           : 文字列変数S $を出力
```

【実行結果】

```
1. 0    2. 0    3. 0    4. 0    5. 0
6. 0    7. 0    8. 0    9. 0   10. 0
11. 0   12. 0   13. 0   14. 0   15. 0
16. 0   17. 0   18. 0   19. 0   20. 0
A B C D E F G I H J K L M N O P Q R S T
```

(12) D I S A B L E I N T R

【概要】

ON (KEY/SRQ/ISRQ) 命令 ((42), (43)参照) によって生じた割り込みの受信を禁止します。

【書式】

DISABLE INTR

【解説】

- ENABLE INTR命令による割り込み受信許可状態を禁止状態にします。

【注意】

- 割り込みによる分岐が生じた場合、分岐先で割り込みが入れ子 (nest) にならないように分岐先でDISABLE INTR命令を実行して下さい。
- 割り込み処理をサブ・ルーチンにした場合は、サブ・ルーチン内のRETURN命令の直前に、ENABLE INTR命令 ((13)参照) を入れると円滑に処理ができます。

【プログラム例】

INTEGER S	: 整数型変数宣言
ON ISRQ GOSUB *SWPEND	: 割り込み (ISRQ) 分岐先定義
OUTPUT 31;"IP SWISC SO SI"	: IP、掃引1秒、割り込み出力設定、シングル掃引
*L	: (ループ)
GOSUB *ONESWP	: 1回掃引サブ・ルーチン
M=MAX(0,700,0)	: 最大レベル (ビルトイン関数)
PRINT M	: 最大レベル出力
GOTO *L	: ラベル *Lへジャンプ
!	:
*ONESWP	:
ENABLE INTR	: 割り込み受信許可
OUTPUT 31;"SI"	: シングル掃引スタート
F=0	: F=0
*LL	: (ループ)
IF F THEN RETURN	: Fが真ならばリターン
GOTO *LL	: ラベル *LLへジャンプ
!	:
*SWPEND	:
DISABLE INTR	: 割り込み受信禁止
S=SPOLL(31)	: 本体計測部へのシリアル・ポール
IF S BAND 4 THEN F=1	: 3ビット目がオンならばF=1を代入
RETURN	: リターン

【実行結果】

1 掃引ごとの最大レベルを出力します。

(13) ENABLE INTR

【概要】

ON (KEY/SRQ/ISRQ) 命令 ((42), (43)参照) によって生じた割り込みの受信を許可します。

【書式】

```
ENABLE INTR
```

【解説】

- ・ DISABLE INTR 命令 ((12)参照) による割り込み受信禁止状態を許可状態にします。

【注意】

- ・ 割り込みによる分岐が生じた場合、分岐先で割り込みが入れ子 (nest) にならないように分岐先で DISABLE INTR を実行して下さい。
- ・ 割り込み処理をサブ・ルーチンにした場合は、サブ・ルーチン内の RETURN 命令の直前に ENABLE INTR 命令を入れると円滑に処理ができます。
- ・ プログラムの RUN 後は、割り込み受信禁止状態になっているので、割り込み処理を使う場合は、ENABLE INTR 命令を実行して下さい。

【プログラム例】

```
ON KEY 1 GOSUB *K1
ON KEY 2 GOSUB *K2
ENABLE INTR
*L
GOTO *L
!
*K1
DISABLE INTR
PRINT "KEY1"
ENABLE INTR
RETURN
*K2
DISABLE INTR
PRINT "KEY2"
ENABLE INTR
RETURN
```

【実行結果】

フル・キーボードの1または2、あるいは本体パネルのテンキーの1、2を押すと、KEY1、KEY2と表示します。

(14) ENTER

【概要】

#ファイル・ディスクリプタに割り当てられているファイルからデータを入力（読み込み）します。

【書式】

ENTER <#ファイル・ディスクリプタ> ; <X> [, <X>]

X: 入力項目（数値変数, 文字列変数）

【解説】

- ・ ファイル・ディスクリプタに割り当てられているファイルからデータを、対応する入力項目のデータ・タイプの形式で読み込み、その入力項目に代入します。
- ・ OPEN命令((44)参照)で指定したタイプにより以下の書式で読み込まれます。

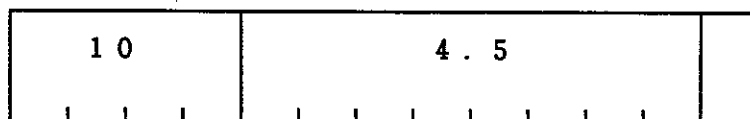
BINARY タイプ

データを内部表現と同じ型で入力（読み込み）します。

整数型 : 4 バイト
実数型 : 8 バイト
文字列 : 4 バイトのヘッダが示すバイト数のデータ

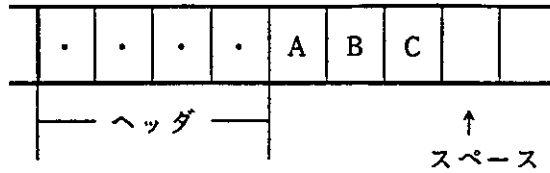
例)

```
INTEGER I  
OPEN "FILE" FOR INPUT AS #FD  
ENTER #FD: I, R, S$
```



変数が実数型のときは、8バイトのデータを読み込み、そのままのデータを変数に代入する。

変数が整数型のときは4バイトを読み込み、そのままのデータを変数に代入する。



変数が文字列のときは、ヘッダを読み込みヘッダが示す長さ分だけ読み込み、文字列変数に代入します。

TEXT タイプ

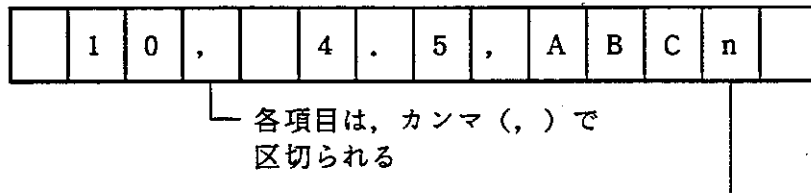
入力項目の数にかかわらずライン・フィードまで読み込みます。カンマ(,)までが1つのデータとなり、入力項目の型に変換して代入されます。

データをASCIIコードに変換して出力します。数値データは、スペースか符号がデータの先頭に付きます。文字列データの最後にはライン・フィード(0x0a)が付きます。

例)

```

INTEGER I
OPEN "FILE" FOR INPUT AS #FD;TEXT
ENTER #FD;I, R, S$
  
```



文字列の最後にはライン・フィード(0x0a)がある

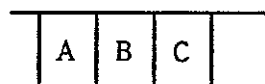
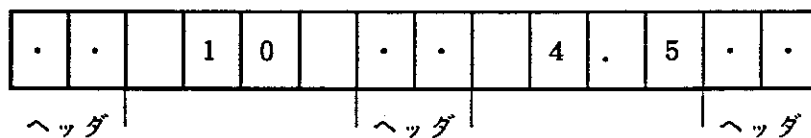
ASCII タイプ

ヘッダ2バイトを読み込み、ヘッダが示す長さのデータを読み込み変数の型にデータを変換し代入します。

例)

```

INTEGER I
OPEN "FILE" FOR INPUT AS #FD;ASCII
ENTER #FD;I, R, S$
  
```



【例】

```
ENTER #FD;S$
```

```
ENTER #FD;A, B
```

【注意】

- ・ ファイル・ディスクリプタは、ファイル・オープン時に指定したものを使います。
- ・ ヘッダとは各項目の区切りとしての目印となり、データの長さを持ちます。
- ・ 入力項目の数が実際のデータより多いときは、多い分の変数には、代入されません。したがって、それらは前に格納されていた値がそのまま残り、逆に変数の数が実際のデータの数よりも少ない場合は、データが読み込まれずに捨てられます。
- ・ 読み込むバイト数は入力項目の型で決まるので、OUTPUTのときと同じ型で入力しないとデータの内容が違ってきます。
- ・ OPEN命令でファイルをオープンしないで本命令を使うとエラーになります。
(file is NOT open)

【プログラム例】

```
OPEN "A" FOR INPUT AS #FA;BINARY  
INTEGER S  
FOR I=1 TO 10  
  ENTER #FA;R,S  
  PRINT R,S  
NEXT I  
CLOSE #FA
```

```
OPEN "AA" FOR OUTPUT AS #FB;TEXT  
INTEGER S  
FOR I=1 TO 10  
  ENTER #FB;R,S  
  PRINT R,S  
NEXT I  
CLOSE #FB
```

```
OPEN "AAA" FOR OUTPUT AS #FC;ASCII  
INTEGER S  
FOR I=1 TO 10  
  ENTER #FC;R,S  
  PRINT R,S  
NEXT I  
CLOSE #FC
```

【実行結果】

上記プログラムを実行すると各プログラムとも以下のようにプリントします。

1. 0	1
2. 0	2
3. 0	3
4. 0	4
5. 0	5
6. 0	6
7. 0	7
8. 0	8
9. 0	9
10. 0	10

(15) ENTER (ENT と省略可)

【概要】

GPIB、本器測定部、またはパラレルI/Oからデータを取り込みます。


【書式】


ENTER <装置アドレス> | 31 | 32 ; <X>

装置アドレス 0～30 : GPIB接続の機器からのデータ入力
31 : 本器の計測部からのデータ入力
32 : パラレルI/Oからのデータ入力

X : [数値変数 | 文字列変数 {, 数値変数 | 文字列変数}]

【解説】

- 装置アドレス 0～30は、背面パネルのGPIB（下側：CONTROLLER）上にある指定された装置からデータを変数に入力します。
- 31を設定すると本器の計測部から内部メモリを通してデータを変数に入力します。
- 32を設定すると16ビットパラレルI/Oからデータを変数に入力します。 32指定時は数値変数のみになります。文字列変数で指定すると、エラーになり、プログラムの実行を中止します。
(invalid type in assnum)
- 変数をカンマ(,)で区切りことにより変数を複数、指定できます。
(ENTER 32に対しては無効)
この場合、受信したデータがカンマで区切られているごとに変数に代入します。受信したデータより変数が多い場合は、データが割当たらない変数には、何も代入されません。逆にデータ数が多い場合そのデータは無視されます。ENTER命令で受信できる文字は1024文字です。その数より多い場合は無視します。

 次頁に示すパラレルI/Oを参照

接続コネクタ: 36ピン アンフェノールコネクタ (プラグ)

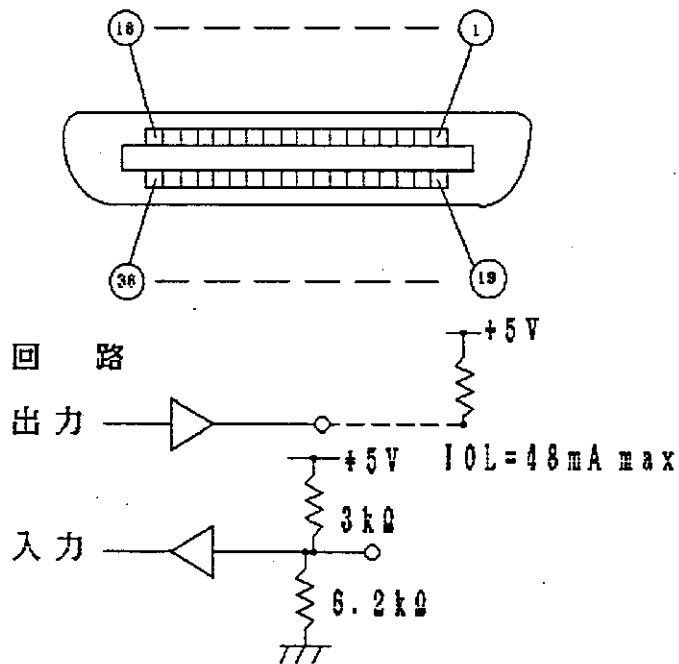
端子番号	信号名	端子番号	信号名
1	GND	19	*OE
2	IN0	20	OUT0
3	IN1	21	OUT1
4	IN2	22	OUT2
5	IN3	23	OUT3
6	IN4	24	OUT4
7	IN5	25	OUT5
8	IN6	26	OUT6
9	IN7	27	OUT7
10	IN8	28	OUT8
11	IN9	29	OUT9
12	IN10	30	OUT10
13	IN11	31	OUT11
14	IN12	32	OUT12
15	IN13	33	OUT13
16	IN14	34	OUT14
17	IN15	35	OUT15
18		36	

LSB

MSB

【36ピン コネクタ端子割当表】

- IN0 ~ IN15 : 入力 (TTL) 正論理
- OUT0 ~ OUT15 : 出力 (TTLオープンコレクタ) 正論理
- *OE : 入力 出力イネーブル 負論理



・ OUTPUT 32 を使用したとき

(2/2)

OUTPUT 32 ; 1 ----> OUT0 が、Hレベルになります。

OUTPUT 32 ; 3 ----> OUT0 と OUT1 が、Hレベルになります。

・ ENTER 32 ; を使用したとき

IN0 が Hレベルのとき ENTER 32 ; A Aは1が代入されます。

IN1

IN2 が Hレベルのとき ENTER 32 ; A Aは6が代入されます。

【例】

```
ENTER 5 ; A
ENTER 5 ; A, B
ENTER 5 ; S $

ENTER 3 1 ; A
ENTER 3 1 ; A, B
ENTER 3 1 ; S $

ENTER 3 2 ; A
```

【注意】

- ・ 選択番号 0 ~ 3 3 以外を指定すると、エラーになり、プログラムの実行を中止します。
(UNIT addr error in ENTER)
- ・ 入力されたデータと変数の型が合わないと、エラーになります。

【プログラム例】

その 1

```
DIM B$[80]
OUTPUT 3;"CF?"
ENTER 3;A
OUTPUT 5;"SP?"
ENTER 5;B$
PRINT A
PRINT B$
```

その 2

```
DIM S$[80]
OUTPUT 31;"CF1. 23MZ"
OUTPUT 31;"CF?"
ENTER 31;A
ENTER 31;S$
PRINT A
PRINT S$
```

その 3

```
INTEGER A
ENTER 32;A
IF A BAND 4 THEN PRINT "BIT 3"
```

(16) F O R . . . T O . . . S T E P
B R E A K / C O N T I N U E ~ N E X T

【概要】

FOR文からNEXT文までの間にある命令を繰り返して実行します。

【書式】

FOR <数値変数> = <初期値> TO <最終値>
[STEP <増分値>]

BREAK

CONTINUE

NEXT <数値変数>

初期値、 最終値、 増分値 : 数値表現式

【解説】

- ・ 指定された<数値変数>をループ（繰り返し）のカウンタとして用い、初期値から最終値まで増分値ずつ変化させていきます。カウンタの値が最終値を超えたとき、ループは終了しNEXT文の次の命令へと移ります。
- ・ カウンタの増減はNEXT文を使います。（数値変数に増分値を加え、FOR文に戻ります。）
- ・ STEP <増分値>を省略した場合、増分値は1となります。
- ・ FOR~NEXTは、入れ子（nest）にできます。
- ・ 一对で使用するFOR~NEXT文の<数値変数>は、同じものでなくてはなりません。変数名が異なっているとエラーになります。
(NEXT without FOR)
- ・ BREAK文を使うと、FOR~NEXTのループから抜け出すことができます。
- ・ CONTINUE文を使うと、NEXT文に達しなくても<数値変数>に増分値を加え、ループ処理を先頭に戻します。

【例】

```
FOR I=1 TO 5 : 1-1
  PRINT I : 1-2
NEXT I : 1-3

FOR J=1 TO 20 STEP 3 : 2-1
  PRINT J : 2-2
  IF J>12 THEN BREAK : 2-3
NEXT J : 2-4

FOR K=10 TO 1 STEP -.5 : 3-1
  IF K>3 THEN CONTINUE : 3-2
  PRINT K : 3-3
NEXT K : 3-4
```

1-1 : カウンタ Iに1を代入、I>5になるまでループ
1-2 : Iをプリント
1-3 : NEXT (1-1に戻る)

2-1 : カウンタ Jに1を代入、ステップ3でJ>20になるまでループ
2-2 : Jをプリント
2-3 : J>12ならばBREAK (このループから抜ける)
2-4 : NEXT (2-1に戻る)

3-1 : カウンタ Kに10を代入、ステップ -.5でK<1になるまでループ
3-2 : K>3ならば、CONTINUE (3-1に戻る)
3-3 : Kをプリント
3-4 : NEXT (3-1に戻る)

【注意】

- FOR~NEXT文でループ処理中に、ループ・カウンタに使用している数値変数の値を変えると、無限ループ等、正常なループの処理を行いません。
- FOR~NEXT文のループ内へGOTO文などで入ってきたり、また逆にループ内から外に抜け出すようなプログラムは、その動作が保証されません。
- BREAK、CONTINUE文は、FOR~NEXT文中でのみ使用できます。
- 以下の場合は、FOR~NEXTが実行されずに、NEXT文の次へ実行が移ります。(実行されなくても、数値変数には初期値が代入されます。)
 - 増分値が正の値で、<初期値>が<最終値>より大きい場合
 - 増分値が負の値で、<初期値>が<最終値>より小さい場合

【プログラム例】

上記【例】のプログラム

【実行結果】

1のプログラム

1. 0
2. 0
3. 0
4. 0
5. 0

2のプログラム

1. 0
4. 0
7. 0
10. 0
13. 0

3のプログラム

3. 0
2. 5
2. 0
1. 5
1. 0
0. 5
0. 0

(17) G L I S T

【概要】

GPIBプリンタにプログラム・リストを出力します。

【書式】

GLIST [<ラベル> [,]]

【解説】

- ・ GPIBプリンタに、パラメータで指定した位置のプログラム・リストを出力します。また、パラメータを省略するとプログラムの先頭から最終行までを出力します。
- ・ ラベルを指定し、ラベルのあとにカンマ (,) を指定すると、そのラベル位置からプログラムの最後までを表示します。
- ・ ラベルの文字数は、最大20文字を目安にしてください。
- ・ プリンタへの出力は、1度BASICプログラムを転送し(MOVE and RUN)、プログラムを停止させ、BASIC modeで(51) PRINTER命令、GLIST命令を行って下さい。なお、プログラム中に置くこともできます。

【例】

```
GLIST
```

```
GLIST *ABC,
```

【注意】

- ・ 行番号指定でも構いませんが書式が上記と異なります。

```
GLIST [<表示開始行>] [, [<表示終了行>]]
```

例)

```
GLIST 100
```

```
GLIST 100,
```

```
GLIST , 200
```

```
GLIST 100, 200
```

- ・ 接続するGPIBコネクタは、背面パネルのCONTROLLER側を使います。
- ・ プリンタのGPIBアドレス指定は、(51) PRINTER命令を参照して下さい。
- ・ 装置アドレスが間違っていたり、指定したアドレスに装置が接続されていない場合には、本命令を無視して次へ進みます。
- ・ プログラム・リストの出力は、BASICインタプリタ側のバッファの最後に転送(MOVE and RUN)されたものが出力されます。

【プログラム例】

その1

リスト出力させたいプログラムをエディタ上に入力し、ポップアップ・メニューの **MOVE and RUN** を選択します。

転送が終了しプログラムの実行が開始されたら、コントロールCでプログラムの実行を停止させます。(最後まで実行させてもよい。)

EASIC modeのミニ・ウィンドウが画面右下に表示されたら(ポップアップ・メニューで選択してもよい)、**PRINTER**命令と**GLIST**命令を実行します。

例)

```
PRINTER 3
GLIST
```

その2

リスト出力させたいプログラムの先頭にあらかじめ

PRINTER命令

GLIST命令

STOP命令

を入力しておき、**MOVE and RUN**を実行します。

例)

```
PRINTER 3
GLIST
STOP
.
. 以下プログラム
.
```

【実行結果】

その1

GPIBアドレス3のプリンタにリストを出力します。

その2

MOVE and RUNを実行すると、GPIBアドレス3のプリンタにリストを出力します。

(18) G L I S T N

【概要】

GPIBプリンタにプログラムLISTを出力します。

【書式】

GLISTN [<ラベル>] [, <行数>]

【解説】

- ・ GPIBプリンタに、パラメータで指定した位置のプログラム・リストを出力します。
- ・ ラベルを指定し、ラベルのあとにカンマ(,) 行数を指定すると、そのラベル位置から行数分を表示します。
- ・ ラベルの文字数は、最大20文字を目安にして下さい。
- ・ ラベルを省略し、<行数>を指定すると<行数>が正の値だとプログラムの先頭から行数分、負の値ならばプログラムの最終行から行数分、表示されます。
- ・ 出力の方法は、(17) GLIST命令と同じです。

【例】

```
GLISTN
GLISTN *ABC, 10
GLISTN *ABC, -10
GLISTN , 20
GLISTN , -20
```

【注意】

- ・ 行番号指定でも構いませんが、書式が上記と異なります。

GLISTN [<表示開始行>] [, [<行数>]]

例)

```
GLISTN 100
GLISTN 100, 15
GLISTN 100, -15
GLISTN , 20
GLISTN , -20
```


- プリンタのGPIBアドレス指定は、(51) PRINTER命令を参照して下さい。
- 装置アドレスが間違っていたり、指定したアドレスに装置が接続されていない場合には、本命令を無視して次へ進みます。
- 本命令の使用方法は(17)のGLIST命令と同様です。

(19) G O S U B ~ R E T U R N

【概要】

指定されたサブ・ルーチンへの分岐／復帰を行います。

【書式】

```
G O S U B <ラベル>
```

```
R E T U R N
```

【解説】

- ・ G O S U B <ラベル>で指定されたサブ・ルーチンへ処理の制御を移します。
- ・ ラベルの文字数は、最大20文字を目安にして下さい。
- ・ サブ・ルーチン内のRETURN文よりジャンプしたGOSUB文の次の命令へと処理の制御を戻します。
- ・ G O S U B ~ R E T U R N文は、入れ子 (n e s t) にできるので、サブ・ルーチンから別のサブ・ルーチンへ分岐できます。

注) 入れ子 (n e s t) が多過ぎるとメモリ容量が足りなくなり、エラーになることがあります。

(G O S U B n e s t o v e r f l o w)

【例】

```
G O S U B * S 1
```

```
* S 1
```

```
A = A * 2
```

```
R E T U R N
```

【注意】

- ・ G O S U B <ラベル>でジャンプするときに、その対象となるラベルがないとエラーになり、プログラムの実行を中止します。
(U n d e f i n e d L A B E L)

【プログラム例】

FOR I=2 TO 5	: I に 2 を代入、I > 5 になるまでループ
A=2	: A = 2
B=A	: B = A
GOSUB *SUB1	: サブ・ルーチン *SUB1 へジャンプ
PRINTF "2^%2d = %5d\n\n", I, B	: データを出力
NEXT I	: NEXT
STOP	: プログラム終了
!	:
*SUB1	: ラベル *SUB1
FOR C=1 TO I-1	: C = 2、C > I - 1 になるまでループ
GOSUB *SUB2	: サブ・ルーチン *SUB2 へジャンプ
NEXT C	: NEXT
RETURN	: リターン
!	:
*SUB2	: ラベル *SUB2
B *= A	: B = B * A
RETURN	: リターン

【実行結果】

2 ^ 2 =	4
2 ^ 3 =	8
2 ^ 4 =	16
2 ^ 5 =	32

(20) GOTO

【概要】

指定されたラベルへの分岐します。

【書式】

GOTO <ラベル>

【解説】

- 指定された<ラベル>へ無条件で分岐します。

【例】

```
GOTO *L A
```

【注意】

- GOTO <ラベル>でジャンプするときに、その対象となるラベルがないとエラーになり、プログラムの実行を中止します。
(Undefined LABEL)
- ラベルの文字数は、最大20文字を目安にして下さい。

【プログラム例】

I = 0	:	変数 I に 0 を代入
*L	:	ラベル名
I = I + 1	:	変数 I に 1 を加えて変数 I に代入
IF I = 10 THEN STOP	:	もし、I が 10 になったら終了
PRINT I	:	I を出力
GOTO *L	:	*L にジャンプ

【実行結果】

```
1. 0
2. 0
3. 0
4. 0
5. 0
6. 0
7. 0
8. 0
9. 0
```

(21) G P R I N T

【概要】

GPIBで接続されているプリンタに数値、文字列を出力します。

【書式】

GPRINT [<X> {[, | ; <X>]}] [<;>]

X : 数値表現式 | 文字列表現式

【解説】

- ・ (50)のPRINTER命令で指定されたGPIBアドレスの装置に数値、文字列を出力します。
- ・ 数値表現式、文字列表現式はセミ・コロン (;) またはカンマ (,) で区切って複数にわたり指定できます
- ・ GPRINT文の最後にセミ・コロン (;) を置いた場合は、プリント出力が終わっても改行されません。してがって、次のPRINT文は以前にプリントした行に続いてプリントします。

【例】

```
S$="DEF" : ①  
GPRINT "ABC" ; : ②  
GPRINT S$ : ③  
GPRINT "A=", A : ④  
GPRINT "CF", CFQ, "KHz" : ⑤  
GPRINT A+100 : ⑥
```

- ① 文字列変数S\$に"DEF"を代入
- ② プリンタに"ABC"を改行しないで出力
- ③ 文字列変数S\$を出力
- ④ 文字列定数と、数値変数を出力
- ⑤ 文字列定数と、数値変数を出力
- ⑥ 数値変数に100を加え出力

【注意】

- ・ GPIBプリンタに出力する前は、必ず、PRINTER命令でGPIBプリンタのアドレスを設定しておいて下さい。(GPIBプリンタとのアドレスが合わないとプリンタに出力されません。)
- ・ 装置アドレスが間違っていたり、指定したアドレスに装置が接続されていない場合、本命令を無視して次へ進みます。

【プログラム例】

```
PRINTER 5  
FOR I=1 TO 10  
  GPRINT " I = " ;  
  GPRINT I  
NEXT I
```

【実行結果】

GPIBのアドレス5のプリンタに1から10までと表示します。

(22) G P R I N T U S I N G
(U S E と 省 略 可)

【概要】

数値、文字列などを編集し、GPIBへ出力します。

【書式】

GPRINT USING <イメージ仕様>; {, <X>}

GPRINT USE <イメージ仕様>; {, <X>}

X : 数値表現式 | 文字列表現式

【解説】

- ・ GPIBポートに数値、文字列を編集し、ASCIIコードで出力します。
- ・ (50)のPRINTER命令で指定されたGPIBアドレスの装置に数値、文字列を出力します。
- ・ イメージ仕様の指定は、文字列表現式にてイメージ仕様をカンマ(,)で区切って書式を指定します。(最後は自動的に改行します。)

【イメージ仕様一覧】

D	指定フィールドの余った部分にスペースを表示する
Z	指定フィールドの余った部分に0を詰める
K	数値をそのまま表示する
S	常に+/-の符号を付ける
M	-の符号を付けるか値が正のときはスペースを付ける
(小数点)	小数点を表示する
E	指数形式(e、符号、指数)で表示する
H	数値、文字列をそのまま表示するが、小数点(.)がヨーロッパ・タイプ(,)になる
R	ヨーロッパ・タイプの小数点になる
*	指定フィールドの余った部分に*印を表示する
A	1文字、表示する
k	文字列をそのまま表示する
X	スペースを表示
リテラル	書式指定にリテラルを書くときは、\” で囲む
B	数値をASCIIコードとして表示する
@	改ページする
+	表示位置を同じ行の先頭に移動させる
-	表示位置を次の行に移動させる
#	最後に改行を行なわない
n	n桁の精度で出力する。文字列に対して指定すると、出力の値は、その文字列の長さになる

【例】

```
GPRINT USING "DDD.DD" ; 1. 2
```

```
GPRINT USE "ZZZ.ZZ" ; 1. 2
```

【注意】

- ・ GPRINT USING命令の表示は、復改はされません。改行だけ行なうので、復改をGPRINT USINGの中に入れて下さい。

```
GPRINT USING "K,k" 123,"\n"
```

- ・ イメージ仕様で指定した数がパラメータで指定した数より多い場合、エラーになり、プログラムの実行を中止します。
(Unmatched IMAGE-spec in USING)
- ・ GPIBプリンタに出力する前には、必ず、~~PRINTER~~命令でGPIBプリンタのアドレスを設定して下さい。(GPIBプリンタとのアドレスが合わないとプリンタに出力されません。)
- ・ ~~装置アドレスが間違っていたり、指定したアドレスに装置が接続されていない場合、本命令を無視して次へ進みます。~~

【プログラム例】

```
PRINTER 5  
GPRINT USING "DDD.DD,k"; 1.2,"\r"  
GPRINT USE "ZZZ.ZZ,k"; 1.2,"\r"  
GPRINT USE "***.ZZ,k"; 2.4,"\r"  
AS="K,5X,B,DDDRDD,k"  
GPRINT USE AS; 2.34,65,1.2,"\r"
```

【実行結果】

GPIBアドレス5のプリンタに以下のように出力されます。

```
1. 2 0  
0 0 1. 2 0  
* * 1. 2 0  
2. 3 4           A    1, 2 0
```


(23) I F T H E N E L S E /
 E L S E I F / E N D I F

【概要】

条件判断による分岐、または指定された文を実行します。

【書式】

- I F <論理式> T H E N <文>
- I F <論理式> T H E N
 <複文>
E N D I F
- I F <論理式> T H E N
 <複文>
E L S E
 <複文>
E N D I F
- I F <論理式> T H E N
 <複文>
E L S E I F <条件式> T H E N
 <複文>
E L S E I F <条件式> T H E N
 <複文>
 .
 .
E L S E
 <複文>
E N D I F

【解説】

- <論理式>の条件によりプログラムの実行を制御します。<論理式>の結果が0になったときを偽 (false) とし、0以外のときを真 (true) と判断します。
- <論理式>は、比較演算子を用いた論理式以外に数値表現式を書くこともできます。この場合、演算結果が0になったときを偽 (false)、0以外の値はすべて真 (true) と判断します。
- <論理式>の判断が真 (true) のときは、THEN文を実行します。THEN文の後ろには文を続けることができます。(このときは、IF文は1行でしか書けません。)
- THEN以下の文を複数行続けたい場合は、THEN文で一旦区切り、次の行から複数行にわたり文を書き、最後にEND IFを置きます。また、ELSE文を置くと、偽 (false) のための文も書けます。

- ELSE IFを使用すると、<論理式>をいくつも書けます。
- 比較演算子には以下のものがあります。

記号	意味	例
= (または ==)	等しい	X=Y, X==Y
<>	等しくない	X<>Y
<	小さい	X<Y
>	大きい	X>Y
<=	小さいか等しい	X<=Y
>=	大きいか等しい	X>=Y

注) <=、>=の記号は=<、=>にはなりません。

- 論理演算子を使い、比較演算子をつなげることもできます。

NOT AND OR XOR

【例】

```

IF A=1 THEN PRINT "A=1"

IF B>10 THEN
  PRINT "B>10"
END IF

IF B>2 THEN
  PRINT "B>2"
ELSE
  PRINT "B<=2"
END IF

IF 1<C AND C<5 THEN
  PRINT "1<C<5"
ELSE IF C=10 THEN
  PRINT "C=10"
ELSE
  PRINT "C:ELSE"
END IF

```

【注意】

- IF文を複数行にわたり書く場合、必ずIF文の最後には、END IFを置いて下さい。IF~END IF文の数が一致しないと、エラーになります。
(Unbalanced IF block)

【プログラム例】

FOR I=1 TO 100	: I = 1、I > 100 になるまでループ
IF 10<=I AND I<=20 THEN	: ① 10 以上 20 以下ならば以下の処理
PRINT "10<=I<=20"	: 文字列出力
END IF	: ①のIF文の終わり
IF I=50 OR I=60 THEN	: ② I = 50、または I = 60 ならば以下の処理
IF I=50 THEN	: ③ I = 50 ならば以下の処理
PRINT "I=50"	: 文字列出力
ELSE	: ③ I = 50 以外ならば以下の処理
PRINT "I=60"	: 文字列出力
END IF	: ③のIF文の終わり
ELSE IF I=70 THEN	: ② I = 70 ならば以下の処理
PRINT "I=70"	: 文字列出力
ELSE IF I=90 THEN	: ② I = 90 ならば以下の処理
PRINT "I=90"	: 文字列出力
END IF	: ②のIF文の終わり
NEXT I	: ループ戻り

【実行結果】

```
10<=I<=20
10<=I<=20
10<=I<=20
10<=I<=20
10<=I<=20
10<=I<=20
10<=I<=20
10<=I<=20
10<=I<=20
10<=I<=20
10<=I<=20
10<=I<=20
10<=I<=20
10<=I<=20
10<=I<=20
I=50
I=60
I=70
I=90
```

(24) INITIALIZE
(INITと省略可)

【概要】

メモリ・カードを初期化します。

【書式】

INITIALIZE

INIT

【解説】

- ・メモリ・カードを初期化します。
- ・エディタのBASIC modeにてINITIALIZE命令を実行します。

【例】

INITIALIZE

INIT

【注意】

- ・初期化する際はCAT命令、またはエディタのLoadを実行し、ファイル内容を確かめてから行なって下さい。
- ・新しいメモリ・カードは初期化しないと使えません。また、すでに初期化済みのメモリ・カードを初期化すると、メモリ・カード内のファイルはすべて消去されます。

(25) I N T E G E R

【概要】

変数または配列変数が整数型であることを宣言します。

【書式】

```
INTEGER <X> | <X> (数値表現式) {, <X> |  
                                <X> (数値表現式) }
```

X : 数値変数

【解説】

- INTEGER文で、数値変数、配列変数を指定すると、以後その変数は整数型になります。
- 整数型変数で扱える数値は、整定数で扱える範囲と同じです。

```
2, 147, 483, 648 ~ +2, 147, 483, 647
```
- 整数しか扱わない変数では、INTEGER文で宣言した方が多少、高速に処理できます。
- INTEGER命令で配列宣言すると、指定された大きさの配列変数をメモリ上に確保します。したがって、大きすぎる配列宣言を行なうと、メモリ領域が足りなくなり、エラーとしてプログラムの実行を中止します。
(memory space full)
- 添字を複数個指定すると、個数分の次元を持つ配列変数の指定になります。
(次元数は、メモリ容量が許す限り)

【例】

```
INTEGER A  
INTEGER B, C  
INTEGER D(20)、E(30)
```

【注意】

- ・ FOR文等のループ変数に整数型を使うと高速に処理できます。
- ・ 実数型変数を配列宣言するときは、(C110)のDIM命令を参照して下さい。

【プログラム例】

```
INTEGER A(20),B(5,4),I,X,Y      : 整数型宣言
FOR I=1 TO 20                    : Iが1からI>20になるまでループ
  A(I)=I                          : 配列変数にIを代入
NEXT I                            : NEXT I
FOR X=1 TO 5                      : Xが1からX>5になるまでループ
  FOR Y=1 TO 4                    : Yが1からY>4になるまでループ
    B(X,Y)=A((Y-1)*5+X)          : 配列Bに配列Aを代入
  NEXT Y                          : NEXT Y
NEXT X                            : NEXT X
FOR X=1 TO 5                      : Xが1からX>5になるまでループ
  FOR Y=1 TO 4                    : Yが1からY>4になるまでループ
    CURSOR X*5,Y:PRINT B(X,Y)    : 配列Bを出力
  NEXT Y                          : NEXT Y
NEXT X                            : NEXT X
```

【実行結果】

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20

(26) I N T E R F A C E C L E A R

【概要】

背面パネルのCONTROLLER側のGPIBに接続されているすべての装置のGPIBインタフェースを初期化します。

【書式】

```
INTERFACE CLEAR
```

【解説】

- この命令を実行すると、GPIBの単線信号IFCを約100 μ sの間、出力(true状態)します。
- 本器のGPIBに接続されている装置のすべてのGPIBインタフェースは、IFC信号を受け取ると、トーカー、またはリスナーの状態を解除します。

【例】

```
INTERFACE CLEAR
```

【プログラム例】

```
INTERFACE CLEAR  
OUTPUT 3:"CF1MZ"
```

(27) INPUT (INP と省略可)

【概要】

キーボード、本器パネルから入力したデータを数値変数、または文字列変数に代入します。

【書式】

```
INPUT [<文字列定数>, ] <X> [ , | ; <X>]
```

X : <数値変数> | <文字列変数>

【解説】

- INPUT文を実行すると、プログラムは一時停止して、キー入力待ちになります。キー入力待ちは、Returnキー（ターミナル）、または単位キー（本器パネル）が押されるまで続きます。入力が済み、Returnキーを押すと、キーから入力されたデータが変数に代入されます。
- 文字列定数（プロンプト）を指定すると、画面にプロンプトを出力してから入力待ちになります。
- INPUT文では、数値変数、文字列変数のいずれも使えますが、数値変数を入力しようとしているときに数字以外の文字（英文字、英記号など）を入力すると、その文字は無視されます。もし数字が1文字もないときは、0が変数に代入されます。また、Returnキーのみを押したときには変数への代入は行ないません。つまり、INPUT文の前の値がそのまま残ります。

【例】

```
INPUT A
INPUT " B = ", B
INP C, D$
INP " name & No ? ", NAS, N
```


【注意】

- 文字列を入力するときは、引用符（"）で囲む必要はありません。
- 本体パネルからは文字列の入力はできません。（数値の入力のみ本体パネルのテンキーで行なえます。）

【プログラム例】

```
INPUT "Center Freq. (MHz) ? ", CF
INP "Span Freq. (MHz) ? ", SF
INP "No. ? ", A$
OUTPUT 31;"CF", CF, "MHz"
OUTPUT 31;"SP", SF, "MHz"
PRINT "No. :", A$
PRINT "CF :", CF, "MHz"
PRINT "SF :", SF, "MHz"
```

【実行結果】

```
Center Freq. (MHz) ? 12
Span Freq. (MHz) ? 30
No. ? 5
No. : 5
CF : 12.0 MHz
SF : 30.0 MHz
```

(28) L I S T

【概要】

外部端末または本体の画面にプログラムLISTを表示します。

【書式】

LIST [<ラベル> [,]]

【解説】

- ・ 画面にパラメータで指定した位置のプログラム・リストを表示します。
- ・ ラベルを指定し、その後にカンマ(,)を指定すると、そのラベル位置からプログラムの最後までを表示します。
- ・ リストの出力は、一度BASICプログラムを転送し(MOVE and RUN)、プログラムを停止させ、BASIC modeでLIST命令を行なって下さい。なお、プログラム中に置くこともできます。
- ・ ラベルの文字数は、最大20文字を目安にして下さい。

【例】

LIST

LIST *ABC,

【注意】

- ・ 行番号指定でも構いませんが書式が上記と異なります。

LIST [<表示開始行>] [, [<表示終了行>]]

例)

```
LIST 100
LIST 100,
LIST , 200
LIST 100, 200
```

- ・ 電源ON時に外部端末が接続されていると外部端末に出力されます。接続されていない場合は本体画面に出力されます。

【プログラム例】

エディタのMOVE and RUNを実行し、コントロールCにてプログラムを停止させ、エディタのBASIC modeでLISTを実行します。

【実行結果】

画面にリストを出力します。

(29) LISTN

【概要】

外部端末、または本体の画面にプログラムLISTを表示します。

【書式】

LISTN [<ラベル>] [, <行数>]

【解説】

- ・ 画面にパラメータで指定した位置のプログラム・リストを表示します。
- ・ ラベルを指定し、ラベルのあとにカンマ(,) 行数を指定すると、そのラベル位置から行数分を表示します。
- ・ ラベルを省略し、<行数>を指定すると、<行数>が正の値ならばプログラムの先頭から行数分、負の値ならばプログラムの最終行から行数分表示されます。
- ・ 出力の方法は(28)のLIST命令と同じです。
- ・ ラベルの文字数は、最大20文字を目安にして下さい。

【例】

```
LISTN
LISTN *ABC, 10
LISTN *ABC, -10
LISTN , 20
LISTN , -20
```

【注意】

- ・ 行番号指定でも構いませんが書式が上記と異なります。

LISTN [<表示開始行>] [, [<行数>]]

例)

```
LISTN 100
LISTN 100, 15
LISTN 100, -15
LISTN , 20
LISTN , -20
```

- ・ 電源ON時に外部端末が接続されていると外部端末に出力されます。接続されていない場合は本体画面に出力されます。

(30) L L I S T

【概要】

シリアルI/Oポート (RS-232) に接続されている装置にプログラムLISTを表示します。

【書式】

LLIST [<ラベル> [,]]

【解説】

- ・ シリアルI/Oポート (RS-232) にパラメータで指定した位置のプログラム・リストを表示します。
- ・ ラベルを指定し、その後にカンマ (,) を指定すると、そのラベル位置からプログラムの最後までを表示します。
- ・ リストの出力は、一度BASICプログラムを転送(MOVE and RUN)してプログラムを停止させ、BASIC modeでLLIST命令を行って下さい。なお、プログラム中に置くこともできます。
- ・ ラベルの文字数は、最大20文字を目安にして下さい。

【例】

LLIST

LLIST *ABC,

【注意】

- ・ 行番号指定でも構いませんが書式が上記と異なります。

LLIST [<表示開始行>] [, [<表示終了行>]]

例)

```
LLIST 100
LLIST 100,
LLIST , 200
LLIST 100, 200
```

- ・ シリアルI/Oポート (RS-232) の設定は、(8)のCONTROL命令を参照して下さい。

【プログラム例】

エディタのMOVE and RUNを実行し、コントロールCにてプログラムを停止させ、エディタのBASIC modeでLLISTを実行します。

【実行結果】

画面にリストを出力します。

(31) L L I S T N

【概要】

シリアルI/Oポート (RS-232) に接続されている装置にプログラム LISTを表示します。

【書式】

LLISTN [<ラベル>] [, <行数>]

【解説】

- ・ シリアルI/Oポート (RS-232) にパラメータで指定した位置のプログラム・リストを表示します。
- ・ ラベルを指定し、ラベルの後にカンマ (,) 行数を指定すると、そのラベル位置から行数分を表示します。
- ・ ラベルを省略し、<行数>を指定すると、<行数>が正の値ならばプログラムの先頭から行数分表示されます。負の値ならばプログラムの最終行から行数分表示されず。
- ・ ラベルの文字数は、最大20文字を目安にして下さい。
- ・ 出力方法は、(30) LLIST命令と同じです。

【例】

LLISTN

LLISTN *ABC, 10

LLISTN *ABC, -10

LLISTN , 20

LLISTN , -20

【注意】

- 行番号指定でも構いませんが書式が上記と異なります。

LLISTN [<表示開始行>] [, [<行数>]]

例)

```
LLISTN 100
LLISTN 100, 15
LLISTN 100, -15
LLISTN , 20
LLISTN , -20
```

- シリアルI/Oポート (RS-232) の設定は、(7) CONTROL命令を参照して下さい。

(32) LOCAL

【概要】

指定した装置をリモート状態から解除するか、または、リモート・イネーブル (REN) ラインを偽 (false) にします。

【書式】

LOCAL [装置アドレス {, 装置アドレス}]

装置アドレス: 0~30

【解説】

- 装置アドレスを指定せずにLOCAL命令だけを実行した場合、GPIBのリモート・イネーブル (REN) ラインが偽 (false) となり、GPIB上のすべての装置がローカル状態になります。
RENラインが偽 (false) のときは、OUTPUT命令でのGPIB装置への設定は不可能となります。RENラインを真 (true) にするには(53)のREMOTE命令を実行して下さい。
- LOCAL命令に続いて、装置アドレスを指定した場合、指定した装置アドレスの装置のみ、リモート状態を解除します。装置アドレスはカンマ (,) で区切ると複数指定できます。

【例】

LOCAL

LOCAL 2

LOCAL 3, 4, 5

【プログラム例】

```
REMOTE 11  
WAIT 2000  
LOCAL 11
```

【実行結果】

装置アドレス11をリモート状態にし (リモート・ランプがあれば点灯する)、2秒後にLOCAL状態になります (リモート・ランプが消灯する)。

(33) LOCAL LOCKOUT

【概要】

GPIBに接続されている装置を、パネル面からローカル状態にする機能を禁止します。

【書式】

LOCAL LOCKOUT

【解説】

- GPIB上の装置がリモート状態のとき（本器コントローラにリモート・コントロールされているとき）は、各装置のパネル・キーはロックされ、パネルからデータの設定はできないようになっています。しかし、ローカル・キーだけはロックされずにいます。このキーを押すと各装置は自分自身をローカル状態にして、データの設定を可能にします。このためリモート制御中に種々の障害が生じ、正確なコントロールができなくなってしまう。この時、LOCAL LOCKOUT命令を実行すると、GPIB上の全装置のローカル・キーをロックして、各装置のパネル面からの設定を完全に禁止します。
- LOCAL LOCKOUT命令を実行すると、GPIBユニバーサルコマンドのローカル・ロックアウト（LLO）を送ります。

【例】

LOCAL LOCKOUT

【注意】

- ローカル・ロックアウト状態の解除は、LOCAL命令で行ないます。

【プログラム例】

```
REMOTE 11, 12
LOCAL LOCKOUT

LOCAL
```

【実行結果】

装置アドレス11と12のローカル・キーが効かなくなり、LOCAL命令を実行すると解除できます。

(34) LPRINT

【概要】

RS-232 で接続されている装置（ターミナル、プリンタ等）に数値、文字列を出力します。

【書式】

```
LPRINT [<X> { [ , | ; <X> ] } ]
```

X : 数値表現式 | 文字列表現式

【解説】

- ・ 数値表現式、文字列表現式は、セミ・コロン（;）またはカンマ（,）で区切って複数にわたり指定できます。
- ・ LPRINT文は、必ず改行となります。

【例】

```
S$=" DEF" : ①  
LPRINT " ABC" : ②  
LPRINT S$ : ③  
LPRINT " A=" , A : ④  
LPRINT " CF" , CFQ , " KHz" : ⑤  
LPRINT A+100 : ⑥
```

- ① 文字列変数S\$に" DEF" を代入
- ② " ABC" を改行しないで出力
- ③ 文字列変数S\$を出力
- ④ 文字列と、数値変数を出力
- ⑤ 文字列と、数値変数を出力
- ⑥ 数値変数に100を加え出力

【注意】

- ・ 外部ターミナルを接続して使うときは、LPRINT命令は、(49) PRINT文と同じ働きをします。
- ・ LPRINT文の最後にセミ・コロン（;）を指定しても、改行となります。

【プログラム例】

```
FOR I=1 TO 10      : Iが1からI>10になるまでループ
  LPRINT " I=", I  : IをシリアルI/Oポートに出力
NEXT I             : NEXT I
```

【実行結果】

外部端末が接続されていると、以下のように出力されます。

```
I =          1. 0
I =          2. 0
I =          3. 0
I =          4. 0
I =          5. 0
I =          6. 0
I =          7. 0
I =          8. 0
I =          9. 0
I =         10. 0
```

(35) LPRINT USING
(USE と省略可)

【概要】

数値、文字列などを編集し、RS-232へ出力します。

【書式】

LPRINT USING <イメージ仕様>; {, <X>}

LPRINT USE <イメージ仕様>; {, <X>}

X: 数値表現式 | 文字列表現式

【解説】

- ・ シリアルI/Oポート (RS-232) に数値、文字列を編集しASCIIコードで出力します。
- ・ イメージ仕様の指定は、文字列表現式にてイメージ仕様をカンマ(,)で区切って書式を指定します。(最後は自動的に改行します。)

【イメージ仕様一覧】

D	指定フィールドの余った部分にスペースを表示する
Z	指定フィールドの余った部分に0を詰める
K	数値をそのまま表示する
S	常に+/-の符号を付ける
M	-の符号を付けるか値が正のときはスペースを付ける
(小数点)	小数点を表示する
E	指数形式(e、符号、指数)で表示する
H	数値、文字列をそのまま表示するが、小数点(.)がヨーロッパ・タイプ(,)になる
R	ヨーロッパ・タイプの小数点になる
*	指定フィールドの余った部分に*印を表示する
A	1文字、表示する
k	文字列をそのまま表示する
X	スペースを表示する
リテラル	書式指定にリテラルを書くときは、\" で囲む
B	数値をASCIIコードとして表示する
@	改ページする
+	表示位置を同じ行の先頭に移動させる
-	表示位置を次の行に移動させる
#	最後に改行しない
n	n桁の精度で出力する。文字列に対して指定すると、出力の値は、その文字列の長さになる

【例】

```
LPRINT USING "DDD.DD" ; 1.2
```

```
LPRINT USE "ZZZ.ZZ" ; 1.2
```

【注意】

- ・ LPRINT USING命令の表示は、復改はされません。改行だけ行なうので復改をLPRINT USING中に入れて下さい。

```
LPRINT USING "K,k" 123,"\n"
```

- ・ イメージ仕様で指定した数がパラメータで指定した数より多い場合エラーになりプログラムの実行を中止します。
(Unmatched IMAGE-spec in USING)

【プログラム例】

```
LPRINT USING "DDD.DD,k"; 1.2, "\r"  
LPRINT USE "ZZZ.ZZ,k"; 1.2, "\r"  
LPRINT USE "***.ZZ,k"; 2.4, "\r"  
A$="K,5X,B,DDDRDD,k"  
LPRINT USE A$; 2.34,65,1.2, "\r"
```

【実行結果】

```
1.20  
001.20  
**1.20  
2.34      A      1,20
```

(36) OFF END

【概要】

ON END命令で定義した分岐先を解除します。

【書式】

OFF END <#ファイル・ディスクリプタ>

【解説】

- ON END命令で<#ファイル・ディスクリプタ>に定義した分岐先を解除します。

【例】

OFF END #FD

【注意】

- この命令によって解除したあとに、ENTER #命令を使用してEOFを読み込むとエラーになり、プログラムの実行を中止します。
- 分岐先の定義は、(40) ON END命令を使います。

(37) OFF ERROR

【概要】

ON ERROR命令で定義した分岐先を解除します。

【書式】

OFF ERROR

【解説】

- ON ERROR命令で定義した分岐先を解除します。

【例】

OFF ERROR

【注意】

- この命令によって解除したあとに、プログラムがエラーを起こすと、実行を中止します。
- 分岐先の定義は、(41) ON ERROR命令を使います。

(38) OFF KEY

【概要】

ON KEY命令で定義した分岐先を解除します。

【書式】

OFF KEY <キー番号>

キー番号：数値表現式（1, 2, 3, 4, 5, 6のいずれか）

【解説】

- ・ ON KEY命令で<キー番号>に定義した分岐先を解除します。
- ・ キー番号による分岐先は、それぞれ個別に定義を解除できます。

【例】

```
OFF KEY 1
OFF KEY 2
OFF KEY 3
OFF KEY 4
OFF KEY 5
OFF KEY 6
```

【注意】

- ・ この命令によって解除したあとに、キーを押しても分岐しません。
- ・ 分岐先の定義は、(42) ON KEY命令を使います。

【プログラム例】

ON KEY 1 GOSUB *K1	: キー番号 1 の分岐先を定義 (サブ・ルーチン)
ON KEY 2 GOTO *K2	: キー番号 2 の分岐先を定義 (ジャンプ)
!	
ENABLE INTR	: 割り込み受信許可
*L	: ループ
GOTO *L	: "
!	
*K1	: キー番号 1 の割り込み処理
PRINT "KEY 1"	: "KEY 1" を出力
OFF KEY 1	: キー番号 1 の分岐先の定義を解除
ON KEY 2 GOTO *K2	: キー番号 2 の分岐先を定義 (ジャンプ)
RETURN	: リターン
!	
*K2	: キー番号 2 の割り込み処理
PRINT "KEY 2"	: "KEY 1" を出力
OFF KEY 2	: キー番号 2 の分岐先の定義を解除
ON KEY 1 GOSUB *K1	: キー番号 1 の分岐先を定義 (サブ・ルーチン)
GOTO *L	: *L へジャンプ

【実行結果】

```
KEY 1
KEY 2
KEY 1
KEY 2
.
.
.
```

同じキーを 2 回押しても、同じものが出力されません。

(39) OFF SRQ / ISRQ

【概要】

ON SRQ命令、またはON ISRQ命令で定義した分岐先を解除します。

【書式】

OFF SRQ

OFF ISRQ

【解説】

ON SRQ命令、またはON ISRQ命令で定義した分岐先を解除します。

【例】

OFF SRQ

OFF ISRQ

【注意】

- ・ この命令によって解除したあとに、ENTER #命令を使用してEOFを読み込むとエラーになり、プログラムの実行を中止します。
- ・ 分岐先の定義は、(4.3) ON SRQ命令、またはON ISRQ命令を使います。

(40) ON END ~ GOTO / GOSUB

【概要】

ファイルのEOF (End Of File) 発生時に処理する分岐先を定義します。

【書式】

ON END <#ファイル・ディスクリプタ> GOTO <ラベル>

ON END <#ファイル・ディスクリプタ> GOSUB <ラベル>

【解説】

- ・ ENTER #~ 命令ファイルを読み込みで、ファイルの終わり (EOF) を検出したときに分岐する分岐先を定義します。
- ・ ON END文にてEOF時の処理を定義しないで次々にデータを読み込むと、EOF読み込み時にエラーとなり、実行を中止します。
(end of <ファイル名> file)
- ・ ラベルの文字数は、最大20文字を目安にして下さい。

【例】

```
ON END #FD GOTO *ERR
```

```
ON END #FD GOSUB *ERR
```

【注意】

- ・ 分岐は、EOFが読み込まれたときに行なわれます。また、GOSUB文での定義で分岐した場合の戻り先は、EOFが読み込まれたENTER命令の次の命令からの実行になります。
- ・ ENABLE INTR/DISABLE INTR命令に関係なく定義した分岐先にジャンプします。
- ・ 分岐先定義の解除は、(3.6)OFF END命令で行います。
- ・ データの書き込み、読み込みは、(44)のOUTPUT #、(13)のENTER #を参照して下さい。

【プログラム例】

OPEN "FFF" FOR OUTPUT AS #FD	: " F F F" のファイル名で書き込みオープン
FOR I=100 TO 200	: Iを100から200までループ
OUTPUT #FD;I	: Iのデータをセーブ
NEXT I	: NEXT I
CLOSE #FD	
!	
ON END #FR GOTO *LA	: EOFの割り込み分岐先を定義
OPEN "FFF" FOR INPUT AS #FR	: " F F F" のファイル名で読み出しオープン
*LOOP	:
ENTER #FR;N	: ファイルからデータをロード、Nへ代入
PRINT N	: Nを画面出力
GOTO *LOOP	: ラベル *LOOPへジャンプ
!	
*LA	:
CLOSE #FR	: ファイル・クローズ
PRINT "EOF"	: " EOF" を画面出力
STOP	: プログラム終了

【実行結果】

```
100. 0
101. 0
102. 0
.
.      (省略)
.
199. 0
200. 0
EOF
```

(41) ON ERROR ~ GOTO/GOSUB

【概要】

BASICプログラムの実行中にエラーが発生したときの分岐先を定義します。

【書式】

```
ON ERROR GOTO <ラベル>
```

```
ON ERROR GOSUB <ラベル>
```

【解説】

- ・ BASICプログラム実行中のエラー発生時の分岐先を定義します。
- ・ この命令を実行しておくで、エラーが起こったときに指定した<ラベル>位置へプログラムの実行を移します。
- ・ ビルトイン関数実行時のエラー処理に使用できます。
- ・ ラベルの文字数は、最大20文字を目安にしてください。

【例】

```
ON ERROR GOTO *ERR
```

```
ON ERROR GOSUB *ERR
```

【注意】

- ・ 分岐は、エラーが発生したときに行われます。また、GOSUB文での定義で分岐した場合、サブ・ルーチンからの戻り先はエラーが発生した命令の次の命令からの実行になります。
- ・ ENABLE INTR/DISABLE INTR 命令に関係なく定義した分岐先にジャンプします。
- ・ 分岐先定義の解除は、(3.7) OFF ERROR 命令を使います。

【プログラム例】

ON ERROR GOTO *ERR	: エラー割り込み分岐先定義
PRINT "START"	: "START" を画面出力
PRINT 1/0	: 1 ÷ 0 を実行
PRINT "END"	: "END" を出力
STOP	: プログラム終了
*ERR	
PRINT "<<ERROR>>"	: "<<ERROR>>" を出力
PRINT " >> ",ERRM\$(0)	: エラー・メッセージを出力

【実行結果】

```
START
30: 0 divide
<<ERROR>>
>>      30: 0 divide
```

(42) ON KEY ~ GOTO/GOSUB

【概要】

フル・キーボード、または本体パネル（テンキー）からの割り込みを受信したときの分岐先を定義します。

【書式】

```
ON KEY <キー番号> GOTO <ラベル>
```

```
ON KEY <キー番号> GOSUB <ラベル>
```

キー番号：数値表現式（1, 2, 3, 4, 5, 6のいずれか）

【解説】

- フル・キーボード、または本体パネル（テンキー）による割り込みの分岐先を定義します。
- 使用できるキーは、1, 2, 3, 4, 5, 6までの数値キー、ON KEY <キー番号>で定義できるキー番号も1~6までです。
- 外部端末のテンキーを使う前には、必ず外部端末のテンキーの設定を変更して下さい。
- ラベルの文字数は、最大20文字を目安にして下さい。

【例】

```
ON KEY 1 GOTO *K1  
ON KEY 2 GOSUB *K2  
ON KEY 3 GOTO *K3  
ON KEY 4 GOSUB *K4  
ON KEY 5 GOTO *K5  
ON KEY 6 GOSUB *K6
```

【注意】

- 分岐は、割り込みが発生したときに実行していた命令の処理が終了してから行なわれます。また、GOSUB文での定義で分岐した場合、サブ・ルーチンからの戻り先は割り込みが発生したときに実行していた命令の次の命令からの実行になります。
- 割り込みの受信の許可、禁止は、(13) ENABLE INTR、(12) DISABLE INTRを参照して下さい。
- 分岐先定義の解除は、(38) OFF KEY命令で行ないます。

【プログラム例】

```
ON KEY 1 GOTO *SC
ON KEY 2 GOTO *SD
ON KEY 3 GOTO *SE
ON KEY 4 GOSUB *SF
ON KEY 5 GOSUB *SG
ON KEY 6 GOSUB *RA
ENABLE INTR
*LOOP
  GOTO *LOOP
STOP

*SC:BUZZER 261, 300:GOTO *LOOP
*SD:BUZZER 294, 300:GOTO *LOOP
*SE:BUZZER 330, 300:GOTO *LOOP

*SF
  DISABLE INTR
  BUZZER 349, 300
  ENABLE INTR
  RETURN

*SG
  DISABLE INTR
  BUZZER 392, 300
  ENABLE INTR
  RETURN

*SA
  DISABLE INTR
  BUZZER 440, 300
  ENABLE INTR
  RETURN
```

【実行結果】

1, 2, 3, 4, 5, 6 を ド、レ、ミ、ファ、ソ、ラ と対応させて音を出します。

(43) ON SRQ / ISRQ ~
GOTO / GOSUB

【概要】

本器の GPIB コントローラが SRQ (サービス・リクエスト) を受信したときの分岐先を定義します。

【書式】

```
ON SRQ GOTO <ラベル>
ON SRQ GOSUB <ラベル>
(外部 GPIB からの SRQ による分岐先定義)
```

```
ON ISRQ GOTO <ラベル>
ON ISRQ GOSUB <ラベル>
(本体計測部からの SRQ による分岐先定義)
```

【解説】

- SRQ (サービス・リクエスト) による割り込みの分岐先を定義します。
- ON SRQ は、外部 GPIB 装置からの SRQ (サービス・リクエスト) の分岐先を定義します。
- ON ISRQ は、本体計測部からの SRQ (サービス・リクエスト) の分岐先を定義します。
- ラベルの文字数は、最大 20 文字を目安にして下さい。

【例】

```
ON SRQ GOTO *SS
ON SRQ GOSUB *SS
ON ISRQ GOTO *IS
ON ISRQ GOSUB *IS
```

【注意】

- ON ISRQ 命令を使うときには、計測部に SRQ 割り込みを発生させる必要があります。SRQ を発生させるには以下の GPIB コードです。OUTPUT 31 命令 ((46) 参照) にて実行して下さい。

GPIBコード	内 容
S0	計測部からコントローラに対し SRQ 割り込みを送信する
S1	計測部からコントローラに対し SRQ 割り込みを送信しない
S2	ステータス・バイトをクリアする

- ・ 分岐は、割り込みが発生したときに実行していた命令の処理が終了してから行なわれます。また、GOSUB文での定義で分岐した場合、サブ・ルーチンからの戻り先は割り込みが発生したときに実行していた命令の次の命令かの実行になります。
- ・ 割り込みの受信の許可、禁止は、(13) ENABLE INTR、(12) DISABLE INTRを参照して下さい。
- ・ 分岐先定義の解除は、(39) OFF SRQ/ISRQで行ないます。

【プログラム例】

例1)

```
TGT=7
ON SRQ GOSUB *SS
ENABLE INTR
*MAINLOOP
  GOSUB *BEEP
  GOTO *MAINLOOP

*BEEP
BUZZER 440,20
WAIT 200
RETURN

*SS
DISABLE INTR
S=SPOLL(TGT)
PRINT "SPOLL",S
ENABLE INTR
RETURN
```

例2)

```
INTEGER S
ON ISRQ GOTO *SS
*ST
OUTPUT 31;"S0 SI"
ENABLE INTR
*MAINLOOP
  GOTO *MAINLOOP

*SS
S=SPOLL(31)
IF S BAND 4 THEN
  PRINT "MAX :",MAX(0,700,0)
  PRINT "MIN :",MIN(0,700,0)
GOTO *ST
```

【実行結果】

例1)

外部GPIB装置からのSRQ（サービス・リクエスト）により分岐し、シリアル・ボールの結果を出力します。

例2)

本体計測部にて、掃引終了ごとにMAX/MINレベルを求めます。

(44) OPEN

【概要】

ファイルに対しファイル・ディスクリプタに割り当て、指定した処理モードでオープンします。

【書式】

```
OPEN "ファイル名" FOR <処理モード> AS ファイル・  
        ディスクリプタ [ ; <タイプ> ]
```

<処理モード>

OUTPUT	: 書き込み
INPUT	: 読み出し

<タイプ>

BINARY	: バイナリ
TEXT	: ASCIIコード
ASCII	: ヘッダ付きASCIIコード

【解説】

- ・ ファイルをプログラムに認識させるために、ファイルに対してファイル・ディスクリプタを割り当て、指定したモードでオープンします。
- ・ ファイル名の文字数は、最大10文字までです。
- ・ 処理モードには OUTPUT と INPUT があります。
 OUTPUT はファイルにデータを書き込むとき
 INPUT はファイルからデータを読み込むとき
 に使用します。
- ・ #ファイル・ディスクリプタ

実際のファイルに対する読み書きは、ENTER/OUTPUT命令を使いますが、これらの命令に対して対象となるファイルを認識させるためにファイル・ディスクリプタを使用します。

ファイル・ディスクリプタ名は #の後に英数字で最大20文字を目安に記述して下さい。

・ タイプ

タイプには、BINARY, TEXT, ASCIIの3種類があります。(タイプの指定がない場合は、BINARY となります。)

BINARY

データを内部表現のまま読み書きします。

整数のときは 4 バイト

実数のときは 8 バイト

文字列は ヘッダ4バイト+文字列(ASCIIデータ)

文字列が奇数のときはデータの後に1バイトのスペースを埋めます。

TEXT

データをそのままASCIIコードに変換して読み書きします。データが数値のときは、数値の前に符号分として、-かスペースをとります。

TEXTタイプ指定では、USING 指定ができます。

ASCII

ヘッダ(識別子)を2バイトの後にASCIIで表現します。データが数値のときは、数値の前に符号分として、-かスペースをとります。文字列が奇数のときはデータの後に1バイトのスペースを埋めます。

【例】

```
OPEN "FILE0" FOR OUTPUT AS #10P
OPEN "FILE1" FOR OUTPUT AS #FD
OPEN "FILE2" FOR OUTPUT AS #FD;BINARY
OPEN "FILE3" FOR OUTPUT AS #FF;TEXT
OPEN "FILE4" FOR OUTPUT AS #FA;ASCII
```

```
OPEN "FILE0" FOR INPUT AS #10P
OPEN "FILE1" FOR INPUT AS #FD
OPEN "FILE2" FOR INPUT AS #FD;BINARY
OPEN "FILE3" FOR INPUT AS #FF;TEXT
OPEN "FILE4" FOR INPUT AS #FA;ASCII
```

【注意】

- ・ すでに他のファイルに割り当てられているファイル・ディスクリプタをオープンすると、エラーになりプログラムの実行を中止します。
("ファイル名" file is already exist.)
("ファイル名" file is already with another PATH.)
- ・ ファイルをオープンしたときには、最後にクローズを必ず実行して下さい。
- ・ 同じファイルを同時点で複数のファイル・ディスクリプタでオープンできません。
- ・ すでにメモリ・カードに存在するファイルをOUTPUT (書き込み) モードでオープンすると、エラーになりプログラムの実行を中止します。
これは誤って必要なファイルを消してしまうのを防ぐためです。
新しくファイルを作成し直すには、(58) PURGE命令でファイルを削除しておきます。
- ・ 同時に複数のファイルをオープンすることはできません。OUTPUTモードで1つ、INPUTモードで1つのみです。

【プログラム例】

```
OPEN "A" FOR OUTPUT AS #FA: BINARY
FOR I=1 TO 10
  OUTPUT #FA: I
NEXT I
CLOSE #FA
```

```
OPEN "AA" FOR OUTPUT AS #FB: TEXT
FOR I=1 TO 10
  OUTPUT #FB: I
NEXT I
CLOSE #FB
```

```
OPEN "AAA" FOR OUTPUT AS #FC: ASCII
FOR I=1 TO 10
  OUTPUT #FC: I
NEXT I
CLOSE #FC
```

【実行結果】

上記プログラムを実行して CAT命令を実行すると

```
ファイル A      のデータ・サイズ : 000080
ファイル AA     のデータ・サイズ : 000051
ファイル AAA    のデータ・サイズ : 000062
```

となります。

(45) OUTPUT

【概要】

#ファイル・ディスクリプタに割り当てられているファイルにデータを出力（書き込み）します。

【書式】

OUTPUT <#ファイル・ディスクリプタ> ; <X> [, <X>]

X : 出力項目 (数値表現式, 文字列表現式)

【解説】

OPEN命令((44)参照)で指定したタイプにより以下の書式で書き込まれます。

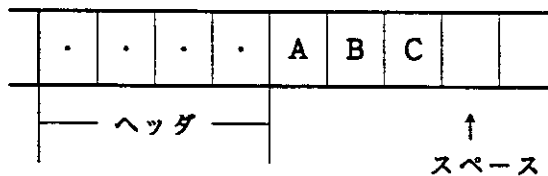
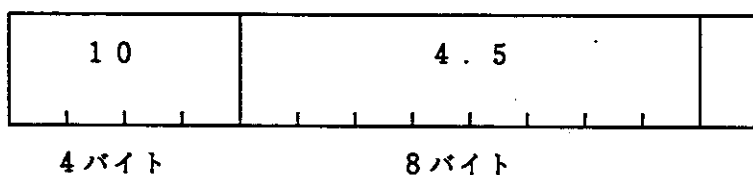
BINARY タイプ

データを内部表現と同じ型で出力（書き込み）します。

整数型 : 4バイト
実数型 : 8バイト
文字列 : 4バイトのヘッダ+文字列数バイト
(ただし、文字列数が奇数の場合は、文字列の最後にスペースを付ける。)

例)

```
OPEN "FILE" FOR OUTPUT AS #FD
OUTPUT #FD: 10, 4.5, "ABC"
```

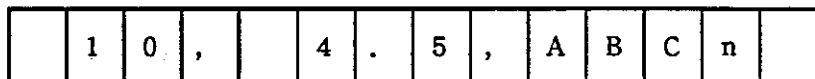


TEXT タイプ

データをASCIIコードに変換して出力します。数値データは、スペースか符号がデータの先頭に付きます。文字列データの最後にはライン・フィード(0x0a)が付きます。

例)

```
OPEN "FILE" FOR OUTPUT AS #FD;TEXT  
OUTPUT #FD;10, 4.5, "ABC"
```



各項目は、カンマ(,)で
区切られる

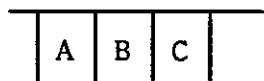
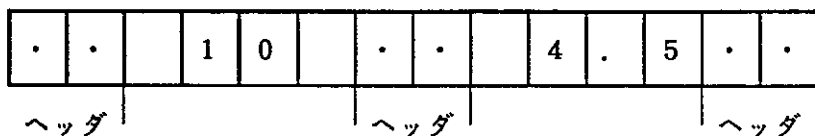
文字列の最後にはライン・フィード
(0x0a)が出力される

ASCII タイプ

データをASCIIコードに変換して出力します。数値データは、スペースか符号がデータの先頭に付きます。
各項目の先頭にはヘッダが2バイト付きます。
データ数が奇数の場合には最後にスペースが付きます。

例)

```
OPEN "FILE" FOR OUTPUT AS #FD;ASCII  
OUTPUT #FD;10, 4.5, "ABC"
```



【例】

```
OUTPUT #FD;"ABC"
```

```
OUTPUT #FD;A, B, C
```

【注意】

- ・ ファイル・ディスクリプタは、ファイル・オープン時に指定したものを 사용합니다。
- ・ オープン時に処理の対象となるファイルに (43) の OPEN #命令でファイル・ディスクリプタを割り当てます。以後のファイルに対する処理はすべてこのファイル・ディスクリプタを介して行ないます。
- ・ ヘッダとは各項目の区切りとしての目印となり、データの長さを持ちます。
- ・ OPEN命令でファイルをオープンしないで本命令を使用すると、エラーになります。(file is NOT open)

【プログラム例】

```
OPEN "A" FOR OUTPUT AS #FA;BINARY
INTEGER S
FOR I=1 TO 10
  S=I
  OUTPUT #FA:I,S
NEXT I
CLOSE #FA
```

```
OPEN "AA" FOR OUTPUT AS #FB;TEXT
INTEGER S
FOR I=1 TO 10
  S=I
  OUTPUT #FB:I,S
NEXT I
CLOSE #FB
```

```
OPEN "AAA" FOR OUTPUT AS #FC;ASCII
INTEGER S
FOR I=1 TO 10
  S=I
  OUTPUT #FC:I,S
NEXT I
CLOSE #FC
```

【実行結果】

上記プログラムを実行して CAT命令を実行すると

```
ファイル A      のデータ・サイズ : 000120
ファイル AA     のデータ・サイズ : 000082
ファイル AAA    のデータ・サイズ : 000104
```

となります。

(46) OUTPUT (OUTと省略可)

【概要】

GPIB、本器測定部、またはパラレルI/Oにデータを送ります。

【書式】

OUTPUT <装置アドレス> | 31 | 32 ; <X>

装置アドレス 0～30 : GPIB接続の機器へデータ入力
31 : 本器の計測部からのデータ入力
32 : パラレルI/Oからのデータ入力

X : [数値変数 | 文字列変数 {, 数値変数 | 文字列変数}]

【解説】

- 装置アドレス(0～30)によって指定された装置へ、数値および文字列をASCIIコードにしてデータを送ります。 ※1

装置アドレス(0～30)は、カンマ(,)で区切ると複数指定できます。

数値表現式と文字列表現式は、カンマ(,)で区切ると混合して使用できます。

なお、REN(リモート・イネーブル)ラインが真(true)のときにOUTPUT命令の0～30を実行すると、装置アドレスで指定された装置は、自動的にリモート状態になります。

- 31を指定しますと本器の計測部へデータを送ります。 ※1
(計測部へはトーカー/リスナのGPIBコードを送ると、外部コントローラを同じように本器を設定できます。
- 32を設定すると16ビットパラレルI/Oへデータを出力します。 ※2
出力できる値は、0～65535までの16ビットです。32の指定では数値表現式のみ使用できます。文字列表現式ではエラーとなり、プログラムの実行を中止します。
(invalid type in getvi)

※1 R3261/3361シリーズの取扱説明書(別冊)にあるGPIBコード一覧を参照して下さい。

※2 次頁に示すパラレルI/Oを参照

接続コネクタ： 36ピン アンフェノールコネクタ (プラグ)

端子番号	信号名	端子番号	信号名
1	GND	19	*OE
2	IN0	20	OUT0
3	IN1	21	OUT1
4	IN2	22	OUT2
5	IN3	23	OUT3
6	IN4	24	OUT4
7	IN5	25	OUT5
8	IN6	26	OUT6
9	IN7	27	OUT7
10	IN8	28	OUT8
11	IN9	29	OUT9
12	IN10	30	OUT10
13	IN11	31	OUT11
14	IN12	32	OUT12
15	IN13	33	OUT13
16	IN14	34	OUT14
17	IN15	35	OUT15
18		36	

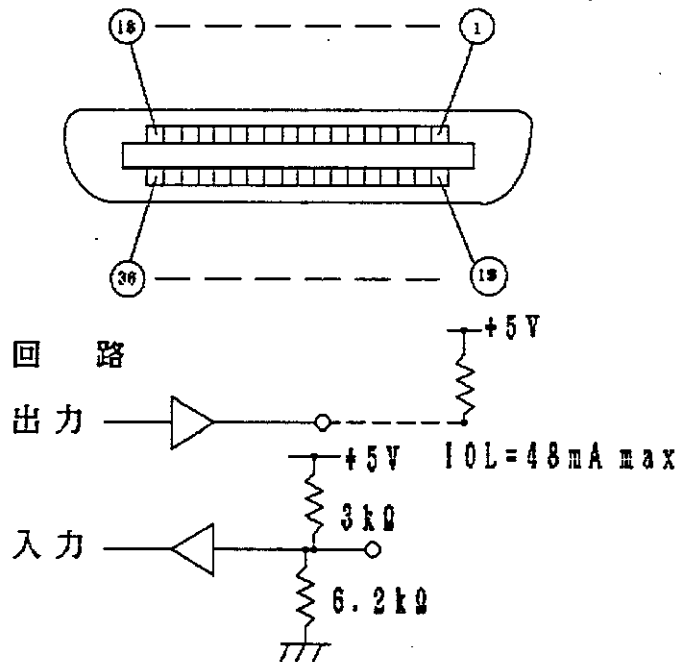
LSB

MSB

【36ピン コネクタ端子割当表】

IN0 ~ IN15 : 入力 (TTL) 正論理
 OUT0 ~ OUT15 : 出力 (TTLオープンコレクタ) 正論理

*OE : 入力 出力イネーブル 負論理



・ OUTPUT 32 を使用したとき

(2/2)

OUTPUT 32 ; 1 ----> OUT0 が、Hレベルになります。

OUTPUT 32 ; 3 ----> OUT0 と OUT1 が、Hレベルになります。

・ ENTER 32 ; を使用したとき

IN0 が Hレベルのとき ENTER 32 ; A Aは1が代入されます。

IN1

IN2 が Hレベルのとき ENTER 32 ; A Aは6が代入されます。

【例】

```
OUTPUT 3 ; " CF3MZ"
OUTPUT 3, 4, 5 ; " SP" , S, " MZ"

OUTPUT 31 ; " CF" , CF, " MZ"

OUTPUT 32 ; 128
OUTPUT 32 ; DT
```

【注意】

- ・ 0～30以外は、カンマ(,)で区切っても指定できません。
(複数のアドレスを指定するのは0～30まで)
- ・ 0～30指定のとき、装置アドレスが間違っていたり、指定したアドレスに装置が接続されていない場合、本命令を無視して次へ進みます。

【プログラム例】

```
OUTPUT 5 ; " CF?"
ENTER 5 ; C
ENTER 5 ; S$
PRINT C, S$

A$ = " CF?"
OUTPUT 31 ; " CF" , CF, " MZ"
OUTPUT 31 ; A$
ENTER 31 ; B$
PRINT B$

INTEGER S
ENTER 32 ; S
IF S BAND 4 THEN OUTPUT 32 ; 128
```

(47) P A U S E

【概要】

プログラムの実行を一時停止させます。

【書式】

P A U S E

【解説】

BASICプログラムの実行をBASICのプログラム自身で一時的に停止するためのコマンドです。一時停止後のプログラムの再開は、(7) CONT命令、またはターミナルの[CONT] (F20) キー、または本器パネルのCONT (ソフト・キー内) で行ないます。

(CONT命令は、BASIC modeで入力して下さい。)

【例】

P A U S E

【プログラム例】

FOR I=1 TO 5	:	カウンタ I を 1 に初期化
PRINT I	:	変数 I を出力
PRINT "Hit CONT key"	:	文字定数を出力
PAUSE	:	PAUSE
NEXT I	:	NEXT

【実行結果】

変数を出力するたびに、一時停止します。

(48) PRINTF (PRF と省略可)

【概要】

外部端末、または本体画面へ数値、文字列を編集し出力します。

【書式】

PRINTF <文字列表現式> [, <X>]

X : 数値表現式 | 文字列表現式

【解説】

- 外部端末使用のときは外部端末画面に出力します。本体のみのときは本体画面に数値、文字列を出力します。
- 文字列表現式に文字列表現式内に以下の書式を指定すると、引数のデータを編集し出力できます。(編集仕様は%で始まります。)

マイナス符号 : 編集されたパラメータをフィールド(出力領域)の左側にそろえることを指定する。

ピリオド : フィールドの幅を示す数字列と桁数(精度)を指定する数字列との区切り。

0 (ゼロ) : 余ったフィールドにゼロ・サプレスを行なう。

- 編集文字とその意味

d : パラメータを10進数に変換する

o : パラメータを符号なし8進数に変換する(先頭に0を付けない)

x : パラメータを符号なし16進数に変換する(先頭に0xを付けない)

s : 文字列

e : 実数として受取り、[-] m. nnnnnnE [±] xxの形の10進数に変換する

(nの文字列の長さは精度によって指定される。標準値は6)

f : 実数として受取り、[-] mmm. nnnnの形の10進数に変換する

(nの文字列の長さは精度によって指定される。標準値は6)

- %の後の文字は、編集文字でなければその文字が出力されます。したがって%%は%%で出力します。

【例】

PRINTF " C=%d", C	: 変数 C を 10 進数で表示
PRINTF " C=%5d", C	: 変数 C を 5 桁の 10 進数で表示 (右詰め)
PRINTF " C=%-5d", C	: 変数 C を 5 桁の 10 進数で表示 (左詰め)
PRINTF " C=%05d", C	: 変数 C を 5 桁の 10 進数で表示 (右詰め: ゼロサプレスする)
PRINTF " H=%x", H	: 変数 H を 16 進数で表示
PRINTF " H=%4x", H	: 変数 H を 4 桁の 16 進数で表示 (右詰め)
PRINTF " H=%-4x", H	: 変数 H を 4 桁の 16 進数で表示 (左詰め)
PRINTF " H=%04x", H	: 変数 H を 4 桁の 16 進数で表示 (右詰め: ゼロサプレスする)
PRINTF " S\$=%s", S\$: 文字列を表示
PRINTF " S\$=% .8s", S\$: 文字列を右詰めで出力する
PRINTF " S\$=%-.8s", S\$: 文字列を左詰めで出力する
PRINTF " %20.10s", S\$: 20 文字のフィールド内で右詰めで 10 文字を表示
PRINTF " %-20.10s", S\$: 20 文字のフィールド内で左詰めで 10 文字を表示
PRINTF " F=%f", F	: 変数 F を 10 進数の実数で表示
PRINTF " F=%8.2f", F	: 変数 F を小数点以下 2 桁の全 8 桁で表 示する。小数点含む
PRINTF " F=%08.2f", F	: 変数 F を小数点以下 2 桁の全 8 桁で表 示する。小数点含む (ゼロ・サプレスする)

【注意】

- PRINTF 文での出力は改行されないので、改行コードを PRINTF 文に入れて使用して下さい。または PRINTF 文の後に PRINT 文を入れて下さい。

```
PRINTF " A=%d\n\r", A
```

改行
- PRINTF の編集文字とパラメータが不足していたり、型指定が間違っていたりすると、無意味な表示となることがあります。

【プログラム例】

```
PRINTF "%d * %d =%05", 12, 34, 12*34
PRINT
PRINTF ":%-10d:%10d:", 123, 456
PRINT
AS="ABCD"
PRINTF ":%8s:%-8s:%8.2s:%-8.2s:", AS, AS, AS, AS
PRINT
PRINTF "%f+%8.3f - %06.3f=%e", 1.23, 4.56, 7.89, 1.23+4.56-7.89
```

【実行結果】

```
12* 34=00408
:123      :      456:
:  ABCD:ABCD  :      AB:AB      :
1.230000+  4.560 - 07.890=-2.100000e+00
```

(49) PRINT (? と省略可)

【概要】

外部端末、または本体画面へ数値、文字列を出力します。

【書式】

```
PRINT [<X> { [ , | ; <X> ] } ] [<, | ; >]
```

X : 数値表現式 | 文字列表現式

【解説】

- ・ 数値表現式、文字列表現式はセミ・コロン (;) またはカンマ (,) で区切って複数にわたり指定できます。(セミ・コロンは、すぐ後に出力、カンマは、タブ間隔で出力されます。)
- ・ PRINT文の最後にセミ・コロン (;) を置いた場合は、出力が終わっても改行されません。してがって、次のPRINT文を実行すると、以前に出力した文字に続いて出力します。

【例】

```
S$="DEF"           : ①  
PRINT "ABC" ;      : ②  
PRINT S$           : ③  
PRINT "A=", A      : ④  
PRINT "CF", CFQ, "KHz" : ⑤  
PRINT A+100        : ⑥
```

- ① 文字列変数S\$に"DEF"を代入
- ② 画面に"ABC"を改行しないで出力
- ③ 文字列変数S\$を出力
- ④ 文字列と、数値変数を出力
- ⑤ 文字列と、数値変数を出力
- ⑥ 数値変数に100を加え出力

【注意】

- ・ 外部端末使用のときは外部端末画面へ、本体のみのときは本体画面に数値、文字列を出力します。

【プログラム例】

```
FOR I=1 TO 10  
  PRINT " I = " ;  
  PRINT I  
NEXT I
```

【実行結果】

```
I = 1. 0  
I = 2. 0  
I = 3. 0  
I = 4. 0  
I = 5. 0  
I = 6. 0  
I = 7. 0  
I = 8. 0  
I = 9. 0  
I = 10. 0
```

外部端末が接続されていれば外部端末に出力されます。接続されていなければ本体画面へ出力されます。

(50) PRINT USING (USE と省略可)

【概要】

数値、文字列などを編集し外部端末、または本体画面へ出力します。

【書式】

```
PRINT USING <イメージ仕様>; {, <X>}
```

```
PRINT USE <イメージ仕様>; {, <X>}
```

X : 数値表現式 | 文字列表現式

【解説】

- 外部端末使用のときは外部端末画面に出力します。本体のみのときは本体画面に数値、文字列を編集し出力します。
- イメージ仕様の指定は、文字列表現式にてイメージ仕様をカンマ(,)で区切って書式を指定します。(最後は自動的に改行します。)

【イメージ仕様一覧】

D	指定フィールドの余った部分にスペースを表示する
Z	指定フィールドの余った部分に0を詰める
K	数値をそのまま表示する
S	常に+/-の符号を付ける
M	-の符号を付けるか値が正のときはスペースを付ける
(小数点)	小数点を表示する
E	指数形式(e、符号、指数)で表示する
H	数値、文字列をそのまま表示するが、小数点(.)がヨーロッパ・タイプ(,)になる
R	ヨーロッパ・タイプの小数点になる
*	指定フィールドの余った部分に*印を表示する
A	1文字、表示する
k	文字列をそのまま表示する
X	スペースを表示する
リテラル	書式指定にリテラルを書くときは、\"で囲む
B	数値をASCIIコードとして表示する
@	改ページする
+	表示位置を同じ行の先頭に移動させる
-	表示位置を次の行に移動させる
#	最後に改行をしない
n	n桁の精度で出力する。文字列に対して指定すると、出力の値は、その文字列の長さになる

【例】

```
PRINT USING "DDD. DD" ; 1. 2
```

```
PRINT USE "ZZZ. ZZ" ; 1. 2
```

【注意】

- PRINT USING命令の表示は、復改はされません。改行だけ行なうので復改をPRINT USINGの中に入れて下さい。

```
PRINT USING "K, k" 1 2 3, "\n"
```

- イメージ

【プログラム例】

```
PRINT USING "DDD. DD, k"; 1. 2, "\r"  
PRINT USE "ZZZ. ZZ, k"; 1. 2, "\r"  
PRINT USE "***. ZZ, k"; 2. 4, "\r"  
AS="K, 5X, B, DDDRDD, k"  
PRINT USE AS; 2. 34, 65, 1. 2, "\r"
```

【実行結果】

```
1. 2 0  
0 0 1. 2 0  
**1. 2 0  
2. 3 4            A    1, 2 0
```

(51) P R I N T E R

【概要】

GPIBプリンタに出力するためのGPIBアドレスを指定します。

【書式】

PRINTER <装置アドレス>

<装置アドレス>: 0~30 まで

【解説】

- ・ GPRINT, GLIST, GLISTN 等の命令の出力先のGPIBアドレスを指定します。(GPRINT, GLIST, GLISTN 等の命令を実行する前に、必ずPRINTER命令を実行して下さい。)
- ・ 装置アドレスの指定は、0~30までです。(デフォルト値は、31で設定してあるので、使用する際には必ずPRINTER命令で設定を0~30までの装置アドレスを毎回設定して下さい。)

【例】

PRINTER 1

PRINTER 5

【注意】

- ・ デフォルトが31で設定してありますが、OUTPUT/ENTER命令のアドレス31とは関係がありません。
- ・ アドレスが設定されていないと出力されません。

【プログラム例】

PRINTER 1	:	出力先を装置アドレス1に設定
GPRINT " ABCD"	:	アドレス1に" ABCD" を出力
A=12	:	変数Aに12を代入
PRINTER A	:	出力先を装置アドレス12に設
GPRINT " printer=" , A	:	アドレス12に文字列と変数内容を

【実行結果】

2台のGPIBプリンタをそれぞれ1, 12にアドレスを設定します。

アドレス1のプリンタには

ABCD

アドレス12のプリンタには

printer = 12.0

と出力します。

(52) READ DATA / RESTORE

【概要】

DATA文に書かれている定数を変数に代入します。

【書式】

```
READ <X> {, <X>}
      <X> : 数値変数 | 文字列変数
```

```
DATA <Y> {, <Y>}
      <Y> : 数値定数 | 文字列定数
```

```
RESTORE [<ラベル>]
```

【解説】

- DATA文で定義されている数値、文字列をREAD文で指定してある変数へ代入します。
- 最初のREADでは、原則として（RESTORE文で変更されていなければ）、DATA文をプログラムの先頭から順に捜して、最初に発見したDATA文の値を変数へ代入します。その後は、順に対応するDATA文の定数を捜して代入していきます。
- DATA文での文字列の定義は、ダブル・クォーテーション（"）で囲みます。
- RESTORE命令は、READ文で読み込むDATA文の先頭を指定します。（ラベルを指定すると、そのラベルの位置以降にあるDATA文を捜します。ラベルの指定がないとプログラムの先頭からDATA文を捜します。）
- ラベルの文字数は、最大20文字を目安にして下さい。

【例】

```
READ A
READ A, B$

DATA 1, 2, 3, 4
DATA "ABC", 5, "DEF", 6

RESTORE
RESTORE *LA
```

【注意】

- READ文の変数に対してDATA文の指定する定数の数が少ない場合、エラーになります。
(Unmatched DATA's values and READ variable)
- READする変数の型とDATA文に指定してある定数の型が合わない場合、エラーになります。
(Invalid type in getdata)

【プログラム例】

RESTORE *LA	: ラベル *LA以降のDATA文から読み込む指定
READ A, B\$: 数値変数A、文字列変数B\$を読み込む
PRINT A, B\$: 変数内容を画面へ出力
RESTORE *LB	: ラベル *LB以降のDATA文から読み込む指定
READ A, B\$: 数値変数A、文字列変数B\$を読み込む
PRINT A, B\$: 変数内容を画面へ出力
STOP	: プログラム終了
!	
*LB	: ラベル *LA
DATA 2, "B"	: データ
*LA	: ラベル *LB
DATA 1, "A"	: データ

【実行結果】

1. 0	A
2. 0	B

(53) REM (! と省略可)

【概要】

プログラムに注釈文を置きます。

【書式】

REM [<文字列>]

! [<文字列>]

【解説】

- ・ プログラムに注釈文を付けたいときに使います。
- ・ REM文は非実行命令であり、REM文に続く文字はすべて無視します。(すべての文字、数字、記号が使用できます。)

【例】

```
REM   <<< r e m a r k >>>
!
!   ****  ADVANTEST  ****
```

【注意】

- ・ REM文の後ろにはコロン(:)によるマルチ・ステートメントは使用できません。すべて注釈文として見なされます。

【プログラム例】

```
! ***** : 注釈
! ** <<PROGRAM>> ** : 注釈
! ***** : 注釈
REM : 注釈
REM          ADVANTEST : 注釈
! : 注釈
PRINT " TEST" : 文字列を出力
STOP : プログラム終了
```

【実行結果】

```
TEST
```


(54) REMOTE

【概要】

指定した装置をリモート状態にするか、または、リモート・イネーブル (REN) ラインを真 (true) にします。

【書式】

REMOTE [装置アドレス {, 装置アドレス}]

装置アドレス: 0~30

【解説】

- ・ 装置アドレスを指定せずにREMOTE命令だけを実行した場合、GPIBのリモート・イネーブル (REN) ラインが真 (true) となりGPIBに接続された装置をリモート・コントロール可能な状態にします。
RENラインを偽 (false) にするためには、(31)のLOCAL命令を実行して下さい。
- ・ REMOTE命令に続いて、装置アドレスを指定した場合、指定した装置アドレスの装置のみ、リモート状態にします。装置アドレスはカンマ (,) で区切ると複数指定できます。
- ・ 以下の命令を実行すると、REMOTE命令を実行しなくてもリモート状態にします。

OUTPUT (46) 参照
SEND LISTEN (61) 参照

【例】

REMOTE

REMOTE 6

REMOTE 6, 7, 10

【プログラム例】

```
REMOTE 11  
WAIT 2000  
LOCAL 11
```

【実行結果】

装置アドレス11をリモート状態 (リモート・ランプが点灯) にして、2秒後にLOCAL状態 (リモート・ランプが消灯) になります。

(55) R E N A M E

【概要】

メモリ・カード内のファイル名を変更します。

【書式】

RENAME <旧ファイル名>, <新ファイル名>

【解説】

- ・ メモリ・カード内のプログラムファイル、データファイルのファイル名を変更します。
- ・ エディタの `BASIC mode` で RENAME 命令を実行します。
- ・ 変更指定されたファイル名が見つからないと、エラーになります。
(RENAME (旧ファイル名, 新ファイル名) error)

【例】

RENAME "BAS1", "BASIC1"

【注意】

- ・ 変更する前には、CAT命令、またはエディタの Load を実行し、変更するファイルを確認して下さい。
- ・ ファイル名は10文字以内です。11文字以上の場合は11文字以降の文字は無視します。
- ・ メモリ・カードのライト・プロテクトがオンのときは実行できません。

(56) REQUEST

【概要】

標準 GPIB (背面パネル上側) に接続されている外部コントローラに SRQ (サービス・リクエスト) を出力します。

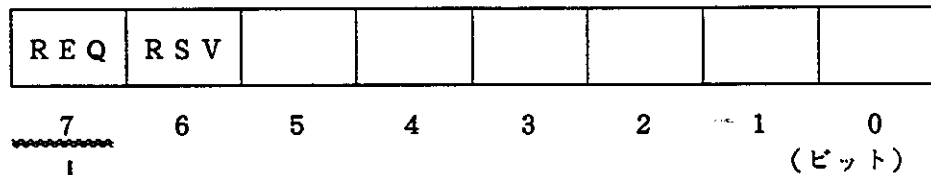
【書式】

REQUEST

【解説】

- ・ 本器コントローラから標準 GPIB (背面パネル上側) に接続されている外部コントローラに SRQ (サービス・リクエスト) を出力するときに使います。
- ・ SRQ は以下のようになります

ステータス・バイト



この命令を実行するとステータス・バイト 7 ビット目がオンになる。

【例】

REQUEST

【プログラム例】

```
IF A>1 THEN
  REQUEST
ELSE
  PRINT "A=1"
END IF
```

【実行結果】

A が A > 1 ならば外部コントローラに SRQ を送出し、データを出力します。それ以外ならば A をプリントします。

(57) RUN

【概要】

BASICプログラムの実行を開始させます。

【書式】

RUN [<ラベル>]

【解説】

- ・ ラベルを省略すると、プログラムの先頭から実行を開始します。
- ・ ラベルを指定すると、そのラベルの位置からの実行を開始します。
- ・ この命令は、**MOVE and RUN**によってBASICバッファに転送済みのプログラムを再度実行(RUNのみ)させます。

【例】

RUN

RUN *ABC

【注意】

- ・ ラベル指定のRUN命令の実行は、BASIC modeで行なって下さい。(プログラムの実行の開始は、外部端末 F19キー、または本体CRTのRUNキーを押して下さい。プログラムはBASICに転送してあるものとします。)
- ・ CONT命令での実行は、変数の初期化をしません。
- ・ ラベルの文字数は、最大20文字を目安にして下さい。

(58) P U R G E

【概要】

メモリ・カード内のファイルを削除します。

【書式】

PURGE <ファイル名>

【解説】

- ・メモリ・カード内のプログラムファイル、データファイルを削除します。
- ・エディタのBASIC modeでPURGE命令を実行します。
- ・削除指定されたファイル名がないときはエラーになります。
(PURGE (ファイル名) error)

【例】

PURGE "BAS1"

【注意】

- ・消去する前には、CAT命令、またはエディタのLoadを実行し、消去するファイルを確認して下さい。
- ・メモリ・カードのライト・プロテクトがオンのときは実行できません。
- ・ファイル名の文字数は、最大10文字までです。

(59) S C R A T C H

【概要】

BASICインタプリタ側の内部バッファを消去します。

【書式】

SCRATCH [<数値表現式>]

<数値表現式> : 1、2 のいずれか

【解説】

- ・すでに転送されているBASICプログラムが不要になったときに、この命令を実行します。
- ・BASIC内部バッファのプログラムのデータ（変数値など）のみを初期化する場合は、<数値表現式>に1を指定します。
- ・BASIC内部バッファのプログラムのみを消去する（変数値は残す）場合は、<数値表現式>に2を指定します。

【例】

SCRATCH

SCRATCH 1

SCRATCH 2

【注意】

- ・BASICプログラムがBASICインタプリタの内部バッファに転送されていないとSCRATCHできません。

(60) S E L E C T C A S E /
 C A S E E L S E /
 E N D S E L E C T

【概要】

1つの式の値を条件として、複数の分岐をします。

【書式】

```
SELECT <数値表現式>
CASE <数値表現式>
END SELECT
```

【解説】

- CASE文に指定された式の値を、SELECT文で指定した式の値が、一致するCASE文のCASE以下の文（複文）を実行します。
- 一致したならばCASE文以下を、一致しなければCASE ELSE文以下を実行します。（CASE ELSEは指定しなくてもよい。）
- SELECT文は入れ子（n e s t）ができます。

【例】

```
INPUT " A= ", A
SELECT A
CASE 1
  PRINT " 1"
CASE 3
  PRINT " 3"
CASE 5
  PRINT " 5"
CASE ELSE
  PRINT " ELSE"
END SELECT
```

【注意】

- SELECT文を指定した場合、必ずEND SELECTを置いて下さい。
- SELECT文は、数値表現式のみ指定できます。

【プログラム例】

INPUT "A= ",A	:	変数Aへ入力
INPUT "B= ",B	:	変数Bへ入力
SELECT A	:	AのSELECT文
CASE 1	:	A = 1 ならば以下の処理
SELECT B	:	BのSELECT文
CASE 10	:	B = 1 0 ならば以下の処理
PRINT "A=1,B=10"	:	文字列出力
CASE ELSE	:	B < > 1 0 ならば以下の処理
PRINT "A=1,B=ELSE"	:	文字列出力
END SELECT	:	BのSELECT文の終了
CASE 2	:	A = 2 ならば以下の処理
PRINT "A=2"	:	文字列出力
CASE 10	:	A = 1 0 ならば以下の処理
PRINT "A=10"	:	文字列出力
END SELECT	:	AのSELECT文の終了

【実行結果】

```

A = 1
B = 1 0
A = 1, B = 1 0

A = 1
B = 2 0
A = 1, B = E L S E

A = 1 0
B = 1 0
A = 1 0

```


(61) SEND

【概要】

GPIB上にコマンド、およびデータを出力します。

【書式】

SEND <A> | | <C> [, <A> |]

A: CMD | DATA | LISTEN [<D> [, <D>]]

B: UNT | UNL

C: TALK [<D>]

D: 数値表現式

【解説】

- GPIB上にユニバーサル・コマンド、アドレス・コマンド、およびデータなどを独立して送る命令です。

CMD : アテンション(ATN)ラインを真(true)にして、与えられた数値を8ビットのバイナリ・データに変換してGPIBに送ります。したがって扱う数値は0~255の範囲内で、また小数点表現の数値は整数に変換されます。
数値を指定しないで実行した場合は、アテンション(ATN)ラインを真(true)にします。

DATA : アテンション(ATN)ラインを偽(false)にして、与えられた数値を8ビットのバイナリ・データに変換してGPIBに送ります。ここで扱う数値は”CMD”で扱われるものと同様です。
数値を指定しないで実行した場合は、アテンション(ATN)ラインを偽(false)にします。

LISTEN : 与えられた数値を、リスナ・アドレス・グループ(LAG)としてGPIB上に送ります。数値は0~30までの数値で複数を指定できます。

TALK : 与えられた数値を、トーカ・アドレス・グループ(TAG)としてGPIB上に送ります。数値は0~30までの数値です。
(複数の指定はできない。)

UNL : アンリスン(UNL)コマンドをGPIB上に送ります。このコマンドを実行する前にリスナに指定されていた装置は、この命令によりリスナ状態が解除されます。

UNT : アントーク(UNT)コマンドをGPIB上に送ります。このコマンドを実行する前にトーカに指定されていた装置は、この命令によりトーカ状態が解除されます。

【例】

```
SEND UNT UNL TALK 1 LISTEN 2,3 DATA 0x43 0x46 0x35 0x4d 0x5a 0xd 0xa  
SEND CMD  
SEND DATA
```

【プログラム例】

```
FOR I=1 TO 9  
  A=I+0x30  
  SEND UNT UNL TALK 30 LISTEN 11 DATA 0x43 0x46 A 0x4d 0x5a 0xd 0xa  
  WAIT 1000  
NEXT I  
SEND UNL UNT
```

【実行結果】

装置アドレス11の中心周波数を1MHz～9MHzまで1秒ごとに設定を変えます。最後に装置アドレス11のリスナ状態を解除します。

(62) SPOLL

【概要】

GPIB上に接続されている指定した特定の装置にシリアル・ボールを行ない、ステータス・バイトを読み込みます。

【書式】

SPOLL (装置アドレス)

装置アドレス 0～30 : GPIB上の装置
31 : 本器計測部

【解説】

- GPIB (CONTROLLER側) に接続されている指定した装置へシリアル・ボールを行ない、ステータス・バイトを読み込みます。
- 本器計測部のステータス・バイトは以下ようになります。

ビット	内 容
0	UNCALを発生したときにこのビットが立つ
1	キャリブレーションが終了したときにこのビットが立つ
2	掃引が終了したときにこのビットが立つ
3	アベレージが設定回数に達したときにこのビットが立つ
4	なし
5	GPIBコードに誤りがあったときにこのビットが立つ
6	SRQ割り込みが発生するモード" S0" に設定されているとき
7	ビット0～5、7のいずれかのビットが立つとこのビットが立つ REQUEST命令を実行したときにこのビットが立つ

【例】

SPOLL (11) : 装置アドレス11にシリアル・ボールを行なう。
SPOLL (31) : 本器計測部へシリアル・ボールを行なう。

【プログラム例】

```
INTEGER S
OUTPUT 31; "S0"
ON ISRQ GOSUB *SS
ENABLE INTR
*L
GOTO *L
*SS
S=SPOLL(31)
IF S BAND 4 THEN PRINT "SWEEP END"
RETURN
```

【実行結果】

本器計測部の掃引終了たびに" SWEEP END" をプリントします。

(63) S P R I N T F

【概要】

数値、文字列を編集し文字列変数に代入します。

【書式】

S P R I N T F <文字列変数>, <文字列表現式> [, <X>]

X : 数値表現式 | 文字列表現式

【解説】

- この命令は、(48) P R I N T F命令とほぼ同じですが、この命令は編集した文字列を指定した文字列変数に代入できます。
- 文字列表現式に文字列表現式内に以下の書式を指定すると、引数のデータを編集し代入できます。(編集仕様は%で始まります。)

マイナス符号 : 編集されたパラメータをフィールド(出力領域)の左側にそろえることを指定する

ピリオド : フィールドの幅を示す数字列と桁数(精度)を指定する数字列との区切り

0(ゼロ) : 余ったフィールドにゼロ・サプレスを行なう

- 編集文字とその意味

d : パラメータを10進数に変換する。

o : パラメータを符号なし8進数に変換する。(先頭に0を付けない)

x : パラメータを符号なし16進数に変換する。(先頭に0xを付けない)

s : 文字列

e : 実数として受取り、[-]m.nnnnnnE[±]xxの形の10進数に変換する。

(nの文字列の長さは精度によって指定される。標準値は6)

f : 実数として受取り、[-]mmm.nnnnnの形の10進数に変換する。

(nの文字列の長さは精度によって指定される。標準値は6)

- %の後の文字は、編集文字でなければその文字が代入されます。したがって%%は%%で代入します。

【例】

- SPRINTF S\$, "C=%d", C
変数Cを10進数で代入する
- SPRINTF S\$, "C=%5d", C
変数Cを5桁の10進数で代入する(右詰め)
- SPRINTF S\$, "C=%-5d", C
変数Cを5桁の10進数で代入する(左詰め)
- SPRINTF S\$, "C=%05d", C
変数Cを5桁の10進数で代入する
(右詰め:ゼロサプレスする)
- SPRINTF S\$, "H=%x", H
変数Hを16進数で代入する
- SPRINTF S\$, "H=%4x", H
変数Hを4桁の16進数で代入する(右詰め)
- SPRINTF S\$, "H=%-4x", H
変数Hを4桁の16進数で代入する(左詰め)
- SPRINTF S\$, "H=%04x", H
変数Hを4桁の16進数で代入する
(右詰め:ゼロサプレスする)
- SPRINTF B\$, "S\$=%s", S\$
文字列を代入する
- SPRINTF B\$, "S\$=%8s", S\$
文字列を右詰めで代入する
- SPRINTF B\$, "S\$=%-8s", S\$
文字列を左詰めで代入する
- SPRINTF B\$, "%20.10s", S\$
20文字のフィールド内で右詰めで10文字を代入する
- SPRINTF B\$, "%-20.10s", S\$
20文字のフィールド内で左詰めで10文字を代入する
- SPRINTF B\$, "F=%f", F
変数Fを10進数の実数で代入する

```
SPRINTF B$, " F=%8. 2 f", F
```

変数 F を小数点以下 2 桁の全 8 桁で代入する (小数点含む)

```
SPRINTF B$, " F=%08. 2 f", F
```

変数 F を小数点以下 2 桁の全 8 桁でゼロ・サプレスして代入する (小数点含む)

【注意】

- SPRINTF の編集文字とパラメータが不足していたり型指定が間違っていたりすると無意味な代入になることがあります。

【プログラム例】

```
DIM SS[80]
SPRINTF SS, "%d * %4d =%05", 12, 34, 12*34
PRINT SS
SPRINTF SS, ":%-10d:%10d:", 123, 456
PRINT SS
AS="ABCD"
SPRINTF SS, ":%8s:%-8s:%8. 2s:%-8. 2s:", AS, AS, AS, AS
PRINT SS
SPRINTF SS, "%f+%8. 3f - %06. 3f=%e", 1. 23, 4. 56, 7. 89, 1. 23+4. 56-7. 89
PRINT SS
```

【実行結果】

```
12* 34=00408
:123      :      456:
:  ABCD:ABCD  :      AB:AB  :
1. 230000+  4. 560 - 07. 890=-2. 100000e+00
```

(64) S T E P

【概要】

BASICプログラムを1行のみ実行させます。

【書式】

STEP [<ラベル>]

【解説】

- ・ ラベルを指定した場合、指定されたラベル位置のプログラムを1行実行します。
- ・ ラベル指定がない場合は、直前に終了した次の行を実行します。
(連続して指定すると1行ずつ実行しながら進みます。)
- ・ PAUSE命令で一時停止しているときにも、STEP命令で1行ずつ進めることができます。

【例】

```
STEP
```

```
STEP *ABC
```

【注意】

- ・ この命令は、ストップ・キー（コントロールC）またはPAUSE命令によってプログラムが停止した状態でのみ実行されます。
(ポップアップ・メニューのBASIC mode内でのみ使用できます。)
- ・ GOTO文、FOR～NEXT文のループの中では、STEP命令は実行しません。
- ・ ラベルの文字数は、最大20文字を目安にして下さい。

(65) S T O P

【概要】

プログラムの実行を中止します。

【書式】

S T O P

【解説】

- プログラムの実行を中止します。S T O P 命令により、プログラムが停止すると以下のメッセージが出力されます。
(Progam ended normally.)

【例】

S T O P

【注意】

- S T O P 命令でプログラムを中止させたとき、C O N T 命令では再実行しません。

【プログラム例】

FOR I=1 TO 10	: カウンタ I に 1 を代入 I > 10 になるまでループ
IF I=5 THEN STOP	: もし、I が 5 ならばプログラム終了
PRINT I	: 変数 I を出力
NEXT I	: ループ戻り

【実行結果】

```
1. 0
2. 0
3. 0
4. 0
Program ended normally.
```


(66) TRIGGER

【概要】

GPIB上に接続されているすべての装置、または指定した特定の装置にアドレス・コマンド・グループ (ACG) のグループ・エグゼキュート・トリガ (GET) を送ります。

【書式】

TRIGGER [装置アドレス {, 装置アドレス}]

装置アドレス: 0~30

【解説】

- ・ 装置アドレスを指定せずにTRIGGER命令だけを実行した場合、GPIBにはグループ・エグゼキュート・トリガ (Group Execute Trigger - GET) のみが送られます。この場合、トリガをかけたい装置はあらかじめリスナに設定されてなければいけません。
- ・ TRIGGER命令に続いて装置アドレスを指定すると、指定した装置にのみGETコマンドを送ります。

【例】

TRIGGER

TRIGGER 2

TRIGGER 3, 4, 5

【プログラム例】

```
TRIGGER 5  
ENTER 5;A$  
PRINT A$
```

【実行結果】

装置アドレス5にトリガをかけ、装置アドレス5からデータを入力します。

(67) WAIT

【概要】

プログラムの実行を指定時間停止させます。

【書式】

WAIT <数値表現式>

【解説】

- ・ 数値表現式は、時間をミリ秒 (msec) で指定します。
- ・ 指定できる時間は、0～63999ミリ秒 (msec) です。

【例】

```
WAIT 1000
WAIT 60000
A=200
WAIT A
```

【注意】

- ・ 時間の指定は、ミリ秒 (msec) 単位です。
- ・ ON (SRQ/ISRQ/KEY) で指定された割り込みが発生してこの命令を実行中は、割り込みによる分岐は行ないません。

【プログラム例】

```
FOR I=1 TO 8
  READ S
  BUZZER S, 1000
  WAIT 1000
NEXT I
DATA 261, 294, 330, 349, 392, 440
DATA 494, 523
```

【実行結果】

”ドレミファソラシド”を1秒ごとに音階を変えて鳴します。

3. ビルトイン関数

3.1 概要

ビルトイン関数を使うと、簡単な操作で解析処理ができソフトウェアの開発時間の短縮、および従来のGPIB操作の処理時間によって短縮できます。以下のようなビルトイン関数が用意されています。

3.2 はじめに

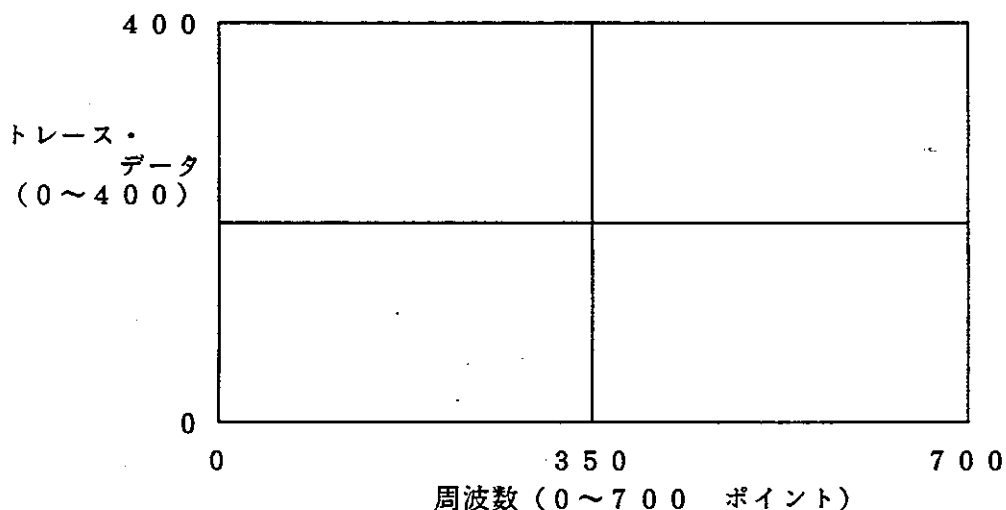
3.2.1 使用開始の前に

(1) ポイントとトレース・データ

ビルトイン関数を使うために、ポイントを理解して下さい。

ポイントには周波数 [横軸] とレベル [縦軸: トレース・データ] があります。これ以降周波数 [横軸] は**ポイント**、レベル [縦軸] は**トレース・データ**とします。

ポイント、トレース・データを使うと、ビルトイン関数が高速に処理でき、高スループットを可能とします。



▲ポイント

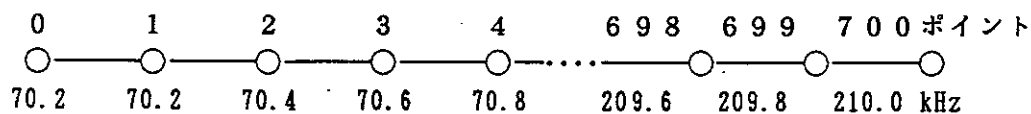
1ポイントあたりの周波数は、 $\text{周波数幅 (SPAN)} \div 700$

▲トレース・データ

1ポイントあたりのレベルは、 $\text{ダイナミック・レンジ} \div 400$

★ ポイント、トレース・データの算出において 700, 400で割り切れないデータは、四捨五入されます。

例 中心周波数 [CENTER] : 140 kHz
周波数幅 [SPAN] : 140 kHz



└ このポイント算出は、四捨五入で左右のポイントに振り分けられる

(2) グラフィック機能

このBASICにはグラフィック機能があります。
グラフィック機能を使用する前に画面の説明をします。

画面は以下の3つのスクリーンがあります。

波形画面 : プリセットを押したときに最初にでる画面
文字画面 (実行結果) : BASICのPRINT命令等で文字が出力される画面
グラフィック画面 : BASICのグラフィック関数で出力される画面

BASICプログラムで各画面へ出力する前に、OUTPUT 31命令で下記のいずれかのGPIBコードを実行して下さい。

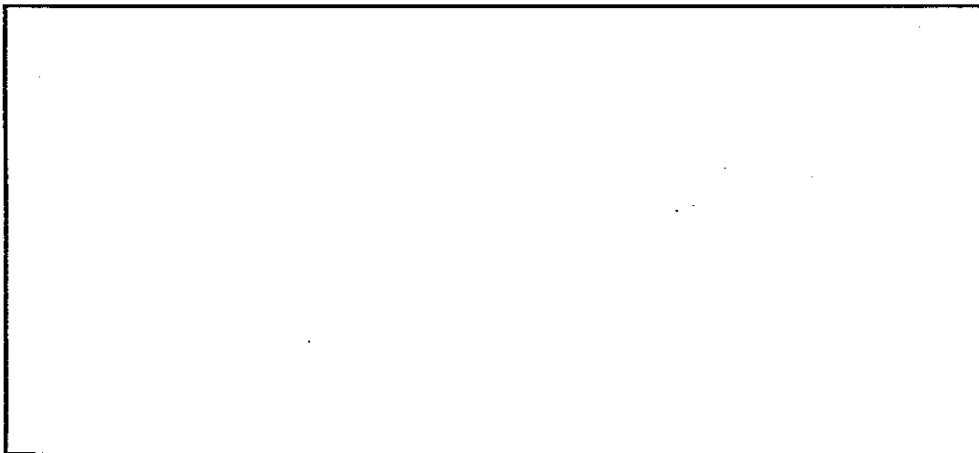
表示状態	GPIBコマンド
波形画面のみ	OUTPUT 31;"VS0"
波形画面+文字画面	OUTPUT 31;"VS1"
文字画面のみ	OUTPUT 31;"VS2"
グラフィック画面	OUTPUT 31;"VS3"

グラフィック関数を使うときには、関数を実行する前にOUTPUT 31;"VS3"を実行して下さい。

グラフィック画面は以下の仕様になっています。

(0,0)

(1023,0)



(0,479)

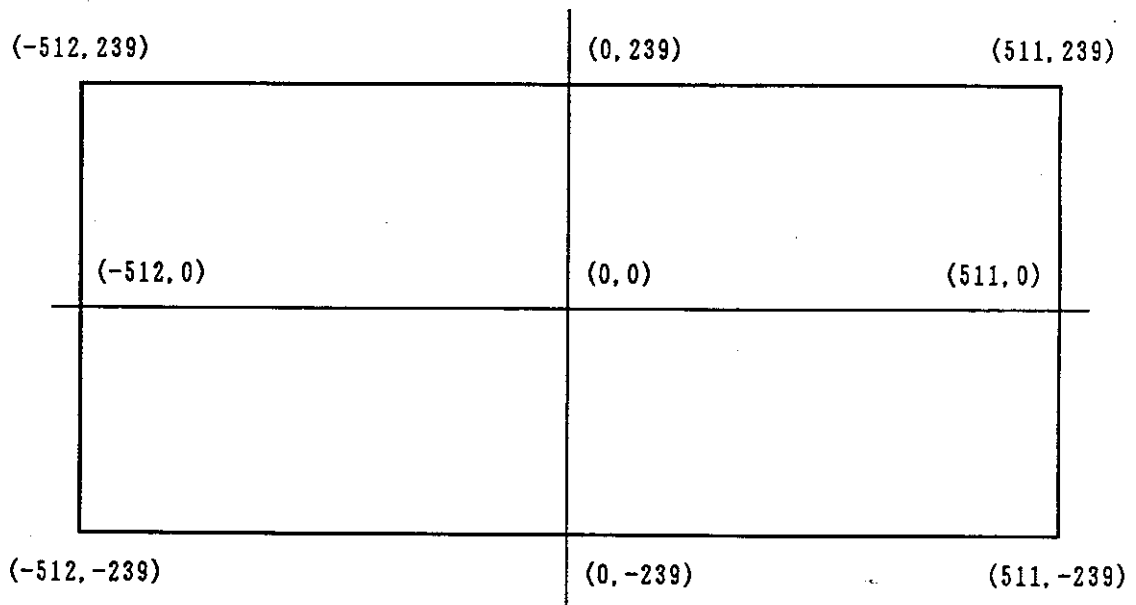
(1023,479)

上図を絶対アドレスと言い、左上が原点(0,0)となります。

GADRS関数にてビューポート・アドレスを設定すると、上図の範囲内なら原点(0, 0)を任意に指定できます。

例) GADRS(1, 512, 240)と設定すると、以下のようになります。

注: ビューポート・アドレス指定時は、設定時の画面範囲を超える指定をすると、エラーになります。



3.4節の(1)～(8)を参照して下さい。

3.2.2 使用上の注意

- ① リップル関係のデータを求めるときは、最初に極大、極小の数を求めて下さい。
以下のビルトイン関数を実行します。実行しないと、エラーになります。(function error)

NRPLH -- 極大のデータが必要なとき
NRPLL -- 極小のデータが必要なとき

注) 極大、極小が必要なときは、両方行ないます。

上記のビルトイン関数を実行してから、以下のビルトイン関数を使って下さい。

PRPLHN
PRPLLN
FRPLHN
FRPLLN
VRPLHN
VRPLLN

- ② 関数のパラメータ等にエラーが生じた場合、エラー・メッセージを出力するだけで、プログラムの実行は停止しません。(function error) エラー時に求まる値は、不定値になります。
- ③ OUTPUT 31, ENTER 31を使って値を求めるよりもビルトイン関数を使用する方が高速に処理できます。(OUTPUT 31, ENTER 31で求めた値も、ビルトイン関数で求めた値も同じ値になります。)
- ④ ポイント(0~700), トレース・データ(0~400), トレースA/B(0/1)の値を、範囲外に指定すると、エラーになります。
(function error)

3.2.3 説明の構成

各関数は、以下の構成で説明します。

FREQ	(周波数)	--	関数名	(戻り値)
【機能】		--	関数の機能について	
【書式】		--	BASICでの構文(書式)	
【戻り値】		--	関数を実行したときに、返る値	
【エラー】		--	設定パラメータの指定時、エラー等について	
【注意】		--	設定パラメータの指定時の注意	
【例】		--	BASICプログラム例	

(1) エラー

- ・ ビルトイン関数にエラーが発生したときは、BASICプログラムの実行は停止しません。プログラムにおけるエラーの認識は、**ON ERROR**文で認識できます。

例

```
ON ERROR GOTO *ERR
```

(2) 機能別ビルトイン関数とグラフィック機能一覧

(1/2)

機 能	関 数	内 容
①周波数を求める	FREQ DFREQ FMAX FMIN BND BNDL BNDH CBND CBNDL CBNDH FRPLHN FRPLL N	ポイント値から周波数を求める ポイント間の周波数を求める ポイント値で指定された2点間の最大レベル位置の周波数を求める ポイント値で指定された2点間の最小レベル位置の周波数を求める ポイント値で指定された位置でのLOSSレベルの帯域幅を求める ポイント値で指定された位置でのLOSSレベル位置での低周波数側を求める ポイント値で指定された位置でのLOSSレベル位置での高周波数側を求める 周波数で指定された位置でのLOSSレベルの帯域幅を求める 周波数で指定された位置でのLOSSレベル位置での低周波数側を求める 周波数で指定された位置でのLOSSレベル位置での高周波数側を求める 周波数軸に対して左からN番目の極大点の周波数を求める 周波数軸に対して左からN番目の極小点の周波数を求める
②ポイントを求める (周波数ポイント :横軸)	POINT DPOINT PMAX PMIN PRPLHN PRPLL N	周波数のポイント(横軸)を求める 指定周波数間のポイント数を求める ポイント値で指定された2点間の最大レベル位置の周波数ポイント(横軸)を求める ポイント値で指定された2点間の最小レベル位置の周波数ポイント(横軸)を求める 周波数軸に対して左からN番目の極大点のポイントを求める 周波数軸に対して左からN番目の極小点のポイントを求める
③ポイントを求める (トレース・データ :縦軸)	LVPOINT LVDPOINT	レベルからトレース・データのポイント(縦軸)値を求める レベル間のトレース・データのポイント(縦軸)値を求める

機 能	関 数	内 容
④レベルを求める	VALUE DVALUE CVALUE DCVALUE LEVEL DLEVEL MAX MIN RPL1 VRPLHN VRPLLN	ポイント値で与えられた周波数位置のレベルを求める ポイント値で与えられた2点の周波数位置のレベル差を求める 周波数位置のレベルを求める 与えられた2点の周波数位置のレベル差を求める ポイント(縦軸)値で与えられたトレース・データのレベルを求める ポイント(縦軸)間でレベルを求める ポイント値で指定された2点間の最大レベルを求める ポイント値で指定された2点間の最小レベルを求める 極大点の最大値と極小点の最小値のレベル差を求める 周波数軸に対して左からN番目の極大点のレベルを求める 周波数軸に対して左からN番目の極小点のレベルを求める
⑤総電力を求める	POWER	ポイント値で指定された2点間の総電力を求める
⑥リップル (極大, 極小) の数を求めます。	NRPLH NRPLL	すべての極大を周波数軸に対して求め, その極大点の数を求める すべての極小を周波数軸に対して求め, その極小点の数を求める
⑦上下限チェックをする	LMTMD1 LMTMD2 LMTUL1 LMTUL2	基準値, 基準値幅, 指定による上下限領域チェックを行なう ポイント値で指定された周波数位置でのレベルを基準値, 基準値幅, 指定により上下限領域チェックを行なう 上下限領域チェックを行なう ポイント値で指定された周波数位置でのレベルの上下限領域チェックを行なう
⑧トレース・データ (read/write)機能	RTRACE WTRACE	指定ポイントのトレース・データを読み込む 指定ポイントのトレース・データを書き込む
⑨グラフィック機能	GADRS GSTR GLINE GSTYLE GPOINT GRECT GFLRECT GMKR	グラフィック・ポイントの絶対アドレス/ビューポートアドレスを指定する 文字列を描く 指定2点間に直線を描く 破線, 点線, 1点鎖線の要素の長さを指定する 指定した位置に点を描く 指定2点間を対角線とする長方形を描く 指定2点間を対角線とする長方形を塗りつぶす 指定した位置にマーカ(ノーマル/デルタ)を描く

(3) パラメータ指定一覧

【ビルトイン、トレース】

```
F =FREQ (P)
F =DFREQ (P1, P2)
F =FMAX (P1, P2, M)
F =FMIN (P1, P2, M)
F =BND (P, X, M)
F1=BNDL (P, X, M)
Fh=BNDH (P, X, M)
F =CBND (F, X, M)
F1=CBNDL (F, X, M)
Fh=CBNDH (F, X, M)
F =FRPLHN (N, M)
F =FRPLL (N, M)

P =POINT (F)
P =DPOINT (F1, F2)
P =PMAX (P1, P2, M)
P =PMIN (P1, P2, M)
P =PRPLHN (N, M)
P =PRPLL (N, M)

T =LVPOINT (L)
T =LVDPOINT (L1, L2)

L =VALUE (P, M)
L =DVALUE (P1, P2, M)
L =CVALUE (F1, F2, M)
L =DCVALUE (F1, F2, M)
L =LEVEL (T)
L =DLEVEL (T1, T2)
L =MAX (P1, P2, M)
L =MIN (P1, P2, M)
L =RPL1 (P1, P2, Dx, Dy, M)
L =VRPLHN (N, M)
L =VRPLL (N, M)

W =POWER (P1, P2, M)

N =NRPLH (P1, P2, Dx, Dy, M)
N =NRPLL (P1, P2, Dx, Dy, M)

C =LMTMD1 (Dd, S, Ds)
C =LMTMD2 (P, S, Ds, M)
C =LMTUL1 (Dd, Up, Lo)
C =LMTUL2 (P, Up, Lo, M)

T =RTRACE (P, M)
WTRACE (T, P, M)
```

注) この関数は値を返しません。

F : 周波数
P : ポイント (0 ~ 700)
L : レベル
T : トレース・データ (0 ~ 400)
M : トレース A / B (0 / 1)
X : loss レベル
C : チェック値 (0, 1, 2, -1)
N : リップル数
W : ワット (電力)

Dx : 横軸微分係数
Dy : 縦軸微分係数
S : 基準値
Ds : 基準値幅
Dd : 被検データ
Lo : 下限値
Up : 上限値

【グラフィック】

```
GADRS (Mo, X, Y)
GFLRECT (D, X1, Y1, X2, Y2)
GLINE (S, D, X1, Y1, X2, Y2)
GMKR (MK, D, X, Y)
GPOINT (D, X, Y)
GRECT (S, D, X1, Y1, X2, Y2)
GSTR (C, X, Y, STR)
GSTYLE (dash, space, dot)
```

C	: 文字サイズ	0	: 16×20ドット
		1	: 18×24ドット
D	: セット/消去	0	: 消去
		1	: セット(描く)
MK	: マーカ	0	: ノーマル・マーカ
		1	: △マーカ
Mo	: アドレス・モード	0	: 絶対アドレス
		1	: ビューポート・アドレス
S	: ライン・スタイル	0	: 実線
		1	: 破線
		2	: 点線
		3	: 一点鎖線
STR	: 文字列表現式		
X	: 座標(横軸)		
Y	: 座標(縦軸)		
dash	: 破線部		
dot	: 点線部		
space	: スペース部		

3.3 各種ビルトイン関数の説明

以下の順（アルファベット順）に説明します。

- (1) BND (周波数)
- (2) BNDL (周波数)
- (3) BNDH (周波数)
- (4) CBND (周波数)
- (5) CBNDL (周波数)
- (6) CBNDH (周波数)
- (7) CVALUE (レベル)
- (8) DCVALUE (レベル)
- (9) DFREQ (周波数)
- (10) DLEVEL (レベル)

- (11) DPOINT (ポイント:横軸 [0~700])
- (12) DVALUE (レベル)
- (13) FMAX (周波数)
- (14) FMIN (周波数)
- (15) FREQ (周波数)
- (16) FRPLHN (周波数)
- (17) FRPLLN (周波数)
- (18) LEVEL (レベル)
- (19) LMTMD1 (チェック値 [0, 1, 2])
- (20) LMTMD2 (チェック値 [0, 1, 2])

- (21) LMTUL1 (チェック値 [0, 1, 2])
- (22) LMTUL2 (チェック値 [0, 1, 2])
- (23) LVDPOINT (トレース・データ: [0~400])
- (24) LVPOINT (トレース・データ: [0~400])
- (25) MAX (レベル)
- (26) MIN (レベル)
- (27) NRPLH (極大点の数)
- (28) NRPLL (極小点の数)
- (29) PMAX (ポイント:横軸 [0~700])
- (30) PMIN (ポイント:横軸 [0~700])

- (31) POINT (ポイント:横軸 [0~700])
- (32) POWER (総電力)
- (33) PRPLHN (ポイント:横軸 [0~700])
- (34) PRPLLN (ポイント:横軸 [0~700])
- (35) RPL1 (レベル)
- (36) RTRACE (トレース・データ: [0~400])
- (37) VALUE (レベル)
- (38) VRPLHN (レベル)
- (39) VRPLLN (レベル)
- (40) WTRCE (トレース・データ: [0~400])

(1) BND (周波数)

【機能】

基準データの測定ポイントと、LOSSレベル、トレースA/Bを指定すると、その指定トレース側での帯域幅を算出します。

【書式】

BND (P, X, M)

P : 基準データのポイント (0~700)

X : LOSSレベル

M : トレース 0 : トレースA

1 : トレースB

【戻り値】

正常終了時 : 指定トレース側の基準データからのLOSSレベルの帯域幅 (Hz)

エラー時 : 不定値

【エラー】

- ・ 指定ポイントが0~700の範囲外の場合、エラーになります。
- ・ エラーのときの戻り値は、不定値になります。
- ・ エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【例】

トレースAのMAX値点からの3dBダウンの帯域幅を求めます。

MP = PMAX (0, 700, 0)

BW1 = BND (MP, 3, 0)

BW2 = BND (PMAX (0, 700, 0), 3, 0)

(BW1とBW2は同じ)

トレースBの中心ポイントから5dBダウンの帯域幅を求めます。

BW = BND (350, 5, 1)

(2) BNDL (周波数)

【機能】

基準データの測定ポイントと、LOSSレベル、トレースA/Bを指定すると、その指定トレース側での帯域幅の低周波数側(左側)の周波数を算出します。

【書式】

BNDL (P, X, M)

P : 基準データのポイント (0~700)

X : LOSSレベル

M : トレース 0 : トレースA

1 : トレースB

【戻り値】

正常終了時 : 指定トレース側の基準データからのLOSSレベルの帯域幅の低周波数側(左側)の周波数(Hz)

エラー時 : 不定値

【エラー】

- ・ 指定ポイントが0~700の範囲外の場合、エラーになります。
- ・ エラーのときの戻り値は、不定値になります。
- ・ エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【例】

トレースAのMAX値点からの5dBダウンの帯域幅の低周波数側(左側)の周波数を求めます。

MP = PMAX (0, 700, 0)

BL1 = BNDL (MP, 5, 0)

BL2 = BNDL (PMAX (0, 700, 0), 5, 0)

(BL1とBL2は同じ)

トレースBの中心ポイントから3dBダウンの帯域幅の低周波数側(左側)の周波数を求めます。

BL = BNDL (350, 3, 1)

(3) BNDH (周波数)

【機能】

基準データの測定ポイントと、LOSSレベル、トレースA/Bを指定すると、その指定トレース側での帯域幅の高周波数側(右側)の周波数を算出します。

【書式】

BNDH (P, X, M)

P : 基準データのポイント (0~700)

X : LOSSレベル

M : トレース 0 : トレースA

1 : トレースB

【戻り値】

正常終了時 : 指定トレース側の基準データからのLOSSレベルの帯域幅の高周波数側(右側)の周波数(Hz)

エラー時 : 不定値

【エラー】

- ・ 指定ポイントが0~700の範囲外の場合、エラーになります。
- ・ エラーのときの戻り値は、不定値になります。
- ・ エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【例】

トレースAのMAX値点からの5dBダウンの帯域幅の高周波数側(右側)の周波数を求めます。

MP = PMAX (0, 700, 0)

BH1 = BNDH (MP, 5, 0)

BH2 = BNDH (PMAX (0, 700, 0), 5, 0)

(BH1とBH2は同じ)

トレースBの中心ポイントから3dBダウンの帯域幅の高周波数側(右側)の周波数を求めます。

BH = BNDH (350, 3, 1)

(4) CBND (周波数)

【機能】

基準データの周波数位置と、LOSSレベル、トレースA/Bを指定すると、その指定トレース側での帯域幅を算出します。

【書式】

CBND (F, X, M)

F : 基準データの周波数

X : LOSSレベル

M : トレース 0 : トレースA

1 : トレースB

【戻り値】

正常終了時 : 指定トレース側の基準データからのLOSSレベルの帯域幅 (Hz)

エラー時 : 不定値

【エラー】

- ・ 指定周波数がスタート周波数からストップ周波数の範囲外の場合、エラーになります。
- ・ エラーのときの戻り値は、不定値になります。
- ・ エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【例】

トレースAの周波数3MHzからの3dBダウンの帯域幅を求めます。

BW = CBND (3 E 6, 3, 0)

トレースBの周波数10MHzからの5dBダウンの帯域幅を求めます。

F = 1 0 E 6

L = 5

BW = CBND (F, L, 1)

(5) CBNDL (周波数)

【機能】

基準データの周波数位置と、LOSSレベル、トレースA/Bを指定すると、その指定トレース側での帯域幅の低周波数側(左側)の周波数を算出します。

【書式】

CBNDL (F, X, M)

F: 基準データの周波数

X: LOSSレベル

M: トレース 0: トレースA

1: トレースB

【戻り値】

正常終了時: 指定トレース側の基準データからのLOSSレベルの帯域幅の低周波数側(左側)の周波数(Hz)

エラー時: 不定値

【エラー】

- 指定周波数がスタート周波数からストップ周波数の範囲外の場合、エラーになります。
- エラーのときの戻り値は、不定値になります。
- エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【例】

トレースAの周波数3MHzからの3dBダウンの帯域幅の左側を求めます。

BW=CBNDL(3E6, 3, 0)

トレースBの周波数10MHzからの5dBダウンの帯域幅の左側を求めます。

F=10E6

L=5

BW=CBNDL(F, L, 1)

(6) CBNDH (周波数)

【機能】

基準データの周波数位置と、LOSSレベル、トレースA/Bを指定すると、その指定トレース側での帯域幅の高周波数側(右側)の周波数を算出します。

【書式】

CBNDH (F, X, M)

F : 基準データの周波数

X : LOSSレベル

M : トレース 0 : トレースA

1 : トレースB

【戻り値】

正常終了時 : 指定トレース側の基準データからのLOSSレベルの帯域幅の低周波数側(右側)の周波数(Hz)

エラー時 : 不定値

【エラー】

- 指定周波数がスタート周波数からストップ周波数の範囲外の場合、エラーになります。
- エラーのときの戻り値は、不定値になります。
- エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【例】

トレースAの周波数3MHzからの3dBダウンの帯域幅の右側を求めます。

BW = CBNDH (3E6, 3, 0)

トレースBの周波数10MHzからの5dBダウンの帯域幅の右側を求めます。

F = 10E6

L = 5

BW = CBNDH (F, L, 1)

(7) CVALUE (レベル)

【機能】

周波数とトレース (A/B) を指定すると、その指定トレース側での指定周波数位置のレベルを求めます。

【書式】

CVALUE (F, M)

F : 指定周波数位置 (Hz)

M : トレース 0 : トレース A

1 : トレース B

【戻り値】

正常終了時 : 指定周波数位置のレベル (単位は、リファレンス・レベルの単位と同じ)

エラー時 : 不定値

【エラー】

- ・ トレース・メモリが、A/Bトレース以外の値、または指定周波数位置がスタート周波数からストップ周波数の範囲外の場合は、エラーになります。
- ・ エラーのときの戻り値は、不定値になります。
- ・ エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【例】

トレース A の 3.5 MHz のレベルを求めます。

```
L = CVALUE ( 3.5 E 6, 0 )
```

トレース B の周波数 3 MHz のレベルを求める。

```
F = 3 E 6  
P = POINT ( F )  
L = VALUE ( P, 1 )
```

$$\left| \begin{array}{l} F = 3 E 6 \\ P = POINT (F) \\ L = VALUE (P, 1) \end{array} \right| = \left| \begin{array}{l} L = CVALUE (3 E 6, 1) \end{array} \right|$$

(左右のプログラムは、同じことを求めています。)

【参考】

(31) POINT

(37) VALUE

(8) DCVALUE (レベル)

【機能】

周波数幅(周波数1、周波数2)の指定と、トレース(A/B)を指定すると、トレース指定側の与えられた2点の周波数位置のレベル差を求めます。

【書式】

DCVALUE (F1, F2, M)

F1 : 指定周波数1 (Hz)

F2 : 指定周波数2 (Hz)

M : トレース 0 : トレースA

1 : トレースB

【戻り値】

正常終了時 : 2点間のレベル差(単位は、dB または mV)

エラー時 : 不定値

【エラー】

- ・ トレース・メモリが、A/Bトレース以外の値、指定周波数1、2がスタート周波数からストップ周波数の範囲外の場合は、エラーになります。
- ・ エラーのときの戻り値は、不定値になります。
- ・ エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【注意】

F1 > F2の場合、F1とF2を入れ換えて処理します。

【例】

トレースAの3MHzと4MHzのレベル差を求めます。

L=DVALUE (3E6, 4E6, 0)

L=DVALUE (4E6, 3E6, 0)

トレースBの2MHzと5MHzのレベル差を求めます。

F=2E6			=	L=DVALUE (2E6, 5E6, 1)
P1=POINT (F)				
P2=POINT (5E6)				
L=DVALUE (P1, P2, 1)				

(左右のプログラムは、同じことを求めています。)

【参考】

(12) DVALUE

(31) POINT

(9) DFREQ (周波数)

【機能】

ポイント幅 (ポイント1, ポイント2) を指定すると、そのポイント幅に相当する周波数幅を算出します。

【書式】

DFREQ (P1, P2)

P1 : 指定ポイント1 (0~700)

P2 : 指定ポイント2 (0~700)

【戻り値】

正常終了時 : 指定ポイント間P1, P2の周波数幅 (Hz)

エラー時 : 不定値

【エラー】

- ・ 指定ポイントが0~700の範囲外の場合、エラーになります。
- ・ エラーのときの戻り値は、不定値になります。
- ・ エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【注意】

P1 > P2 の場合、P1 と P2 を入れ換えて処理します。

【例】

ポイント値 300 と 400 の間の周波数を求めます。

FS = DFREQ (300, 400)

I = 400

FS = DFREQ (I, 300)

(10) DLEVEL (レベル)

【機能】

トレース・データ [縦軸: 0~400] 間 (T1, T2) を指定すると、トレース・データ間に相当するレベルを算出します。

【書式】

DLEVEL (T1, T2)

T1 : 指定トレース・データ1
[縦軸: 0~400]

T2 : 指定トレース・データ2
[縦軸: 0~400]

【戻り値】

正常終了時: トレース・データ [縦軸: 0~400] から換算したレベル (単位は、dB
または mV)

エラー時 : 不定値

【エラー】

- ・ 指定トレース・データが0~400の範囲外の場合、エラーになります。
- ・ エラーのときの戻り値は、不定値になります。
- ・ エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【注意】

T1 > T2 の場合、T1 と T2 を入れ換えて処理します。

【例】

トレース・データ200, 300間のレベルを求めます。

L = DLEVEL (200, 300)

L = DLEVEL (300, 200)

(11) DPOINT (ポイント:横軸 [0~700])

【機能】

周波数幅(周波数1、周波数2)を指定すると、その周波数幅が測定器内部のポイント(0~700)に対して、周波数幅が何ポイントあるかを算出します。

【書式】

DPOINT (F1, F2)

F1 : 指定周波数1 (Hz)

F2 : 指定周波数2 (Hz)

【戻り値】

正常終了時: 指定周波数間F1, F2のポイント数(0~700)

エラー時 : 不定値

【エラー】

- ・ 指定周波数がスタート周波数からストップ周波数の範囲外の場合、エラーになります。
- ・ エラーのときの戻り値は、不定値になります。
- ・ エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【注意】

F1 > F2 の場合、F1 と F2 を入れ換えて処理します。

【例】

周波数3MHzと、4MHzの間のポイント数を求めます。

```
PSPAN=DPOINT (3E6, 4E6)
```

```
PSPAN=DPOINT (4E6, 3E6)
```

周波数3MHzと、3.5MHzの間のポイント数を求めます。

```
FA=3E6
```

```
PSPAN=DPOINT (FA, 3.5E6)
```


(12) DVALUE (レベル)

【機能】

ポイント幅 (ポイント1, ポイント2) の指定と、トレース (A/B) を指定すると、トレース指定側の与えられた2点の周波数位置のレベル差を求めます。

【書式】

DVALUE (P1, P2, M)

P1 : 指定ポイント1 (0~700)

P2 : 指定ポイント2 (0~700)

M : トレース 0 : トレースA

1 : トレースB

【戻り値】

正常終了時 : 2点間のレベル差 (単位は、dB または mV)

エラー時 : 不定値

【エラー】

- ・ トレース・メモリが、A/Bトレース以外の値、または指定ポイントが(0~700)以外の場合は、エラーになります。
- ・ エラーのときの戻り値は、不定値になります。
- ・ エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【注意】

P1 > P2 の場合、P1 と P2 を入れ換えて処理します。

【例】

トレースAの30ポイント目と40ポイント目のレベル差を求めます。

L = DVALUE (30, 40, 0)

L = DVALUE (40, 30, 0)

トレースBの周波数2MHzと5MHzのレベル差を求めます。

F = 2E6			=	L = DCVALUE (2E6, 5E6, 1)
P1 = POINT (F)				
P2 = POINT (5E6)				
L = DVALUE (P1, P2, 1)				

(左右のプログラムは、同じことを求めています。)

【参考】

(8) DCVALUE

(31) POINT

(13) FMAX (周波数)

【機能】

測定ポイント領域(ポイント1, ポイント2)と、トレースA/Bを指定すると、そのトレース側での領域内の縦軸の最大値点の周波数を算出します。

【書式】

FMAX (P1, P2; M)

P1 : 指定ポイント1 (0~700)

P2 : 指定ポイント2 (0~700)

M : トレース 0 : トレースA

1 : トレースB

【戻り値】

正常終了時 : 指定ポイント領域内での縦軸の最大値点の周波数 (Hz)

エラー時 : 不定値

【エラー】

- ・ 指定ポイントが0~700の範囲外の場合、エラーになります。
- ・ エラーのときの戻り値は、不定値になります。
- ・ エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【注意】

P1 > P2 の場合、P1 と P2 を入れ換えて処理します。

【例】

トレースAの 0~700ポイントの間のMAX値点の周波数を求めます。

```
MF = FMAX (0, 700, 0)
```

トレースBの周波数10~20MHz間のMAX値点の周波数を求めます。

```
F1 = 10E6
```

```
P1 = POINT (F1)
```

```
MF1 = FMAX (P1, POINT (20E6), 1)
```

```
MF2 = FMAX (POINT (F1), POINT (20E6), 1)
```

(MF1 と MF2 は 同じ)

【参考】

(31) POINT

(14) FMIN (周波数)

【機能】

測定ポイント領域(ポイント1, ポイント2)と、トレースA/Bを指定すると、そのトレース側での領域内の縦軸の最小値点の周波数を算出します。

【書式】

FMAX (P1, P2, M)

P1 : 指定ポイント1 (0~700)

P2 : 指定ポイント2 (0~700)

M : トレース 0 : トレースA

1 : トレースB

【戻り値】

正常終了時 : 指定ポイント領域内での縦軸の最小値点の周波数 (Hz)

エラー時 : 不定値

【エラー】

- ・ 指定ポイントが0~700の範囲外の場合、エラーになります。
- ・ エラーのときの戻り値は、不定値になります。
- ・ エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【注意】

P1 > P2の場合、P1とP2を入れ換えて処理します。

【例】

トレースAの 0~700ポイントの間のMIN値点の周波数を求めます。

MF = FMIN (0, 700, 0)

トレースBの周波数 10~20MHz間のMIN値点の周波数を求めます。

F1 = 10E6

P1 = POINT (F1)

MF1 = FMIN (POINT (20E6), P1, 1)

MF2 = FMIN (POINT (20E6), POINT (F1), 1)

(MF1 と MF2 は 同じ)

【参考】

(31) POINT

(15) FREQ (周波数)

【機能】

ポイント値を指定すると、そのポイントに相当する周波数を算出します。

【書式】

FREQ (P)

P : ポイント (0 ~ 700)

【戻り値】

正常終了時 : ポイント値を周波数に換算した値 (Hz)

エラー時 : 不定値

【エラー】

- ・ 指定ポイントが0~700の範囲外の場合、エラーになります。
- ・ エラーのときの戻り値は、不定値になります。
- ・ エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【例】

350ポイント目の周波数を求めます。

F = FREQ (350)

400ポイント目の周波数を求めます。

I = 200

F = FREQ (I * 2)

(16) FRPLHN (周波数)

【機能】

トレースA/Bおよび周波数軸に対して、左からN番目の極大点のNo. を指定すると、その極大点の周波数を算出します。

【書式】

FRPLHN (N, M)

N : 周波数軸に対して左からN番目の極大点のNo.

M : トレース 0 : トレースA

1 : トレースB

【戻り値】

正常終了時 : 極大点を周波数軸(横軸 : 左側から)に対して、N番目の極大点の周波数 (Hz)

エラー時 : 不定値

【エラー】

- ・ N番目の極大点がない場合は、エラーになります。
(この関数を使う前に、(27) NRPLH を実行して下さい)
- ・ エラーのときの戻り値は、不定値になります。
- ・ エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【注意】

この関数を使う前には、必ず(27) NRPLH関数を実行してから使用して下さい。

【例】

トレースAの0~700ポイント内に対して、微分係数 $Dx=5$ 、 $Dy=3$ の全極大を求め、左から3番目の極大点の周波数を求めます。

```
RH=NRPLH(0, 700, 5, 3, 0)
F=FRPLHN(3, 0)
```

トレースBの周波数10~20MHz内に対して、微分係数 $Dx=3$ 、 $Dy=5$ の全極大を求め、左から2番目の極大点の周波数を求めます。

```
STF=10E6
SPF=20E6
X=3
Y=5
STP=POINT(STF)
SPP=POINT(SPF)
TS=1
RH=NRPLH(STP, SPP, X, Y, TS)
F=FRPLHN(2, TS)
```

(17) FRPLL (周波数)

【機能】

トレースA/Bおよび周波数軸に対して、左からN番目の極小点のNo. を指定すると、その極小点の周波数を算出します。

【書式】

FRPLL (N, M)

N : 周波数軸に対して左からN番目の極小点のNo.

M : トレース 0 : トレースA

1 : トレースB

【戻り値】

正常終了時 : 極小点を周波数軸(横軸 : 左側から)に対して、N番目の極小点の周波数 (Hz)

エラー時 : 不定値

【エラー】

- N番目の極小点がない場合は、エラーになります。
この関数を使う前に、(28) NRPLL を実行して下さい。
- エラーのときの戻り値は、不定値になります。
- エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【注意】

この関数を使う前には、必ず(28) NRPLL関数を実行して下さい。

【例】

トレースAの0~700ポイント内に対して、微分係数 $Dx=5$, $Dy=3$ の全極小を求め、左から3番目の極小点の周波数を求めます。

```
RH=NRPLL (0, 700, 5, 3, 0)
```

```
F=FRPLL (3, 0)
```

トレースBの周波数10MHz~20MHz内に対して、微分係数 $Dx=3$, $Dy=5$ の全極小を求め、左から2番目の極小点の周波数を求めます。

```
STF=10E6
```

```
SPF=20E6
```

```
X=3
```

```
Y=5
```

```
STP=POINT (STF)
```

```
SPP=POINT (SPF)
```

```
TS=1
```

```
RH=NRPLL (STP, SPP, X, Y, TS)
```

```
F=FRPLL (2, TS)
```

(18) LEVEL (レベル)

【機能】

トレース・データ (縦軸 [0~400]) を指定すると、そのトレース・データに相当するレベルを算出します。

【書式】

LEVEL (T)

T : トレース・データ (縦軸 [0~400])

【戻り値】

正常終了時 : トレース・データ (縦軸 [0~400]) から換算したレベル (単位は、リファレンス・レベルの単位と同じ)

エラー時 : 不定値

【エラー】

- ・ 指定ポイントが0~400の範囲外の場合、エラーになります。
- ・ エラーのときの戻り値は、不定値になります。
- ・ エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【例】

トレース・データ200のレベル値を求めます。

```
L = LEVEL (200)
```

トレースAの全てトレース・データをレベル値に換えます。

```
DIM L [701]
OUTPUT 31 ; "GTA"
FOR I=0 TO 700
  L [I+1] = LEVEL (RTRACE (I, 0))
NEXT I
```

【参考】

(36) RTRACE

(19) LMTMD1 (チェック値 [0, 1, 2])

【機能】

基準値とその基準値の上下限值、および被検データを与えると、その上下限幅の領域内にあるか否かをチェックします。

【書式】

LMTMD1 (D d, S, D s)

D d : 被検データ

S : 基準値

D s : 上下限值

$((S - D s) \leq D d \leq (S + D s))$

【戻り値】

正常終了時 : 領域内のとき -- 0
 上限値より上のとき -- 1
 下限値より下のとき -- 2

【例】

入力した値が、40～60の間にあるかをチェックします。

```
INPUT D
T=LMTMD1 (D, 50, 10)
IF T=0 THEN PRINT "OK"
IF T=1 THEN PRINT "UPPER"
IF T=2 THEN PRINT "LOWER"
```

基準値を入力し、トレースAの30MHzのレベルが、その基準値に対し±10の範囲内であるか否かを判定します。

```
INPUT "STANDARD", S
LV=CVALUE (30E6, 0)
T=LMTMD1 (LV, S, 10)
IF T=0 THEN PRINT "OK"
IF T=1 THEN PRINT "UPPER"
IF T=2 THEN PRINT "LOWER"
```


(20) LMTMD2 (チェック値 [0, 1, 2])

【機能】

基準値と、その基準値の上下限值、およびトレースA/B、ポイントを与えると、そのトレース側のポイントのレベルが上下限幅の領域内にあるか否かをチェックします。

【書式】

LMTMD2 (P, S, Ds, M)

P : ポイント (0~700)

S : 基準値

Ds : 上下限值

M : トレース 0 : トレースA

1 : トレースB

$((S - Ds) \leq \text{レベル} \leq (S + Ds))$

【戻り値】

正常終了時：領域内のとき	--	0
上限値より上のとき	--	1
下限値より下のとき	--	2

【例】

基準値を入力し、トレースAの30MHzのレベルが、その基準値に対し±10の範囲内であるか否かを判定します。

```
INPUT "STANDARD", S
P=POINT(30E6)
T=LMTMD2(P, S, 10, 0)
IF T=0 THEN PRINT "OK"
IF T=1 THEN PRINT "UPPER"
IF T=2 THEN PRINT "LOWER"
```

(21) LMTUL1 (チェック値 [0, 1, 2])

【機能】

上下限值と被検データを与えると、その上下限値の領域内にあるか否かをチェックします。

【書式】

LMTUL1 (Dd, Up, Lo)

Dd : 被検データ

Up : 上限値

Lo : 下限値

(Lo ≤ Dd ≤ Up)

【戻り値】

正常終了時 : 領域内のとき	--	0
上限値より上のとき	--	1
下限値より下のとき	--	2

【例】

入力した値が、30～40の間かをチェックします。

```
INPUT D
T=LMTUL1 (D, 40, 30)
IF T=0 THEN PRINT "OK"
IF T=1 THEN PRINT "UPPER"
IF T=2 THEN PRINT "LOWER"
```

上下限值を入力し、トレースAの30MHzのレベルをチェックします。

```
INPUT "UPPER", U
INPUT "LOWER", L
LV=CVALUE (30E6, 0)
T=LMTUL1 (LV, U, L)
IF T=0 THEN PRINT "OK"
IF T=1 THEN PRINT "UPPER"
IF T=2 THEN PRINT "LOWER"
```

(22) LMTUL2 (チェック値 [0, 1, 2])

【機能】

基準値と、その基準値の上下限值、およびトレースA/B、ポイントを与えると、そのトレース側のポイントのレベルが上下限値の領域内にあるか否かをチェックします。

【書式】

LMTUL2 (P, Up, Lo, M)

P : ポイント (0~700)

Up : 上限値

Lo : 下限値

M : トレース 0 : トレースA

1 : トレースB

(Lo ≤ レベル ≤ Up)

【戻り値】

正常終了時 : 領域内のとき -- 0

上限値より上のとき -- 1

下限値より下のとき -- 2

【例】

上下限值を入力し、トレースAの30MHzのレベルをチェックします。

```
INPUT "UPPER", U
INPUT "LOWER", L
P=POINT(30E6)
T=LMTUL2(P, U, L, 0)
IF T=0 THEN PRINT "OK"
IF T=1 THEN PRINT "UPPER"
IF T=2 THEN PRINT "LOWER"
```

(23) LVDPOINT (トレース・データ: [0~400]).

【機能】

レベル間 (L1, L2) を指定すると、そのレベル間のトレース・データ (縦軸) を算出します。

【書式】

LVDPOINT (L1, L2)

L1: レベル1

(単位は、リファレンス・レベルの単位と同じ)

L2: レベル2

(単位は、リファレンス・レベルの単位と同じ)

【戻り値】

正常終了時: レベルから換算したトレース・データ (縦軸 [0~400])

エラー時: 不定値

【エラー】

- ・ エラーのときの戻り値は、不定値になります。
- ・ エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。
- ・ 指定レベル1, 2が、計測器のスケール内に納まる値である事。

【注意】

L1 > L2 の場合、L1 と L2 を入れ換えて処理します。

【例】

レベル値 -20 dB ~ -30 dB 間のトレース・データを求めます。

TD=LVDPOINT (-20, -30)

レベル値 -20 dB ~ -50 dB 間のトレース・データを求めます。

L=-50

TD=LVDPOINT (-20, L)

(24) LVPOINT (トレース・データ：[0~400])

【機能】

レベルを指定すると、そのレベルに相当するトレース・データ値(縦軸)を算出します。

【書式】

LVPOINT (L)

L : レベル

(単位は、リファレンス・レベルの単位と同じ)

【戻り値】

正常終了時：レベルから換算したトレース・データ(縦軸 [0~400])

エラー時 : 不定値

【エラー】

- ・ エラーのときの戻り値は、不定値になります。
- ・ エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。
- ・ 指定レベルが、計測器のスケール内に納まる値である事。

【例】

レベル値-10dBのトレース・データを求めます。

TD=LVPOINT (-10)

レベル値-20dBのトレース・データを求めます。

L=-20

TD=LVPOINT (L)

(25) MAX (レベル)

【機能】

測定ポイント領域 (ポイント1, ポイント2) と、トレースA/Bを指定すると、そのトレース側での領域内の縦軸の最大値を算出します。

【書式】

MAX (P1, P2, M)

P1 : 指定ポイント1 (0~700)

P2 : 指定ポイント2 (0~700)

M : トレース 0 : トレースA

1 : トレースB

【戻り値】

正常終了時 : 指定ポイント領域内での縦軸の最大レベル (単位は、リファレンス・レベルの単位と同じ)

エラー時 : 不定値

【エラー】

- ・ 指定ポイントが0~700の範囲外の場合、エラーになります。
- ・ エラーのときの戻り値は、不定値になります。
- ・ エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【注意】

P1 > P2の場合、P1とP2を入れ換えて処理します。

【例】

トレースAの 0~700ポイントの間のMAX値を求めます。

ML=MAX (0, 700, 0)

トレースBの周波数 10~20MHz間のMAX値を求めます。

F1=10E6

P1=POINT (F1)

ML1=MAX (P1, POINT (20E6), 1)

ML2=MAX (POINT (F1), POINT (20E6), 1)

(ML1 と ML2は 同じ)

(26) MIN (レベル)

【機能】

測定ポイント領域 (ポイント1, ポイント2) と、トレースA/Bを指定すると、そのトレース側での領域内の縦軸の最小値を算出します。

【書式】

MIN (P1, P2, M)

P1 : 指定ポイント1 (0~700)

P2 : 指定ポイント2 (0~700)

M : トレース 0 : トレースA

1 : トレースB

【戻り値】

正常終了時 : 指定ポイント領域内での縦軸の最小レベル (単位は、リファレンス・レベルの単位と同じ)

エラー時 : 不定値

【エラー】

- ・ 指定ポイントが0~700の範囲外の場合、エラーになります。
- ・ エラーのときの戻り値は、不定値になります。
- ・ エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【注意】

P1 > P2 の場合、P1 と P2 を入れ換えて処理します。

【例】

トレースAの 0~700ポイントの間のMIN値を求めます。

```
ML=MIN (0, 700, 0)
```

トレースBの周波数 10~20MHz間のMIN値を求めます。

```
F1=10E6
```

```
P1=POINT (F1)
```

```
ML1=MIN (POINT (20E6), P1, 1)
```

```
ML2=MIN (POINT (20E6), POINT (F1), 1)
```

(ML1 と ML2 は 同じ)

(27) NRPLH (極大点の数)

【機能】

測定ポイント領域(ポイント1, ポイント2)と、トレースA/B、微分係数(Dx, Dy)を指定すれば、その指定トレース側のポイント領域内において極大を周波数軸(横軸: 左側から)に対して求め、その極大点の数を算出します。

【書式】

NRPLH (P1, P2, Dx, Dy, M)

P1 : 指定ポイント1 (0~700)
P2 : 指定ポイント2 (0~700)
Dx : 微分係数ポイント (0~700)
Dy : 微分係数トレース・データ (0~400)
M : トレース 0 : トレースA
1 : トレースB

【戻り値】

正常終了時 : 極大を周波数軸(横軸: 左側から)に対して求めた、その極大点の数(最大256)
エラー時 : 不定値

【エラー】

- ・ 指定ポイントが0~700の範囲外の場合、エラーになります。
- ・ 極大値および極小値がサーチできない場合は、エラーになります。
(Dx, Dyの数値を変更して下さい)
- ・ エラーのときの戻り値は、不定値になります。
- ・ エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【注意】

P1 > P2の場合、P1とP2を入れ換えて処理します。

PRPLHN, FRPLHN, VRPLHN のビルトイン関数を使う前に、必ずこの関数(NRPLH)を実行して下さい。

【例】

トレースAの0~700ポイント内に対して、微分係数 $Dx=5$, $Dy=3$ の極大の数を求めます。

```
RH=NRPLH (0, 700, 5, 3, 0)
```

トレースBの10~20MHz内に対して、微分係数 $Dx=3$, $Dy=5$ の極大の数を求めます。

```
STF=10E6  
SPF=20E6  
X=3  
Y=5  
STP=POINT (STF)  
SPP=POINT (SPF)  
TS=1  
RH=NRPLH (STP, SPP, X, Y, TS)
```

(28) NRPLL (極小点の数)

【機能】

測定ポイント領域(ポイント1, ポイント2)と、トレースA/B、微分係数(Dx, Dy)を指定すると、その指定トレース側のポイント領域内において、極小を周波数軸(横軸:左側から)に対して求め、その極小点の数を算出します。

【書式】

NRPLL (P1, P2, Dx, Dy, M)

P1 : 指定ポイント1 (0~700)

P2 : 指定ポイント2 (0~700)

Dx : 微分係数ポイント (0~700)

Dy : 微分係数トレース・データ (0~400)

M : トレース 0 : トレースA

1 : トレースB

【戻り値】

正常終了時 : 極小を周波数軸(横軸:左側から)に対して求めた、その極小点の数(最大256)

エラー時 : 不定値

【エラー】

- ・ 指定ポイントが0~700の範囲外の場合、エラーになります。
- ・ 極大値および極小値がサーチできない場合は、エラーになります。
Dx, Dyの数値を変更して下さい。
- ・ エラーのときの戻り値は、不定値になります。
- ・ エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【注意】

P1 > P2の場合、P1とP2を入れ換えて処理します。

PRPLL, FRPLL, VRPLL のビルトイン関数を使う前には、必ずこの関数(NRPLL)を実行して下さい。

【例】

トレースAの0~700ポイント内に対して、微分係数 $Dx=5$, $Dy=3$ の極小の数を求めます。

```
RL=NRPLL (0, 700, 5, 3, 0)
```

トレースBの10~20MHz内に対して、微分係数 $Dx=3$, $Dy=5$ の極小の数を求めます。

```
STF=10E6
```

```
SPF=20E6
```

```
X=3
```

```
Y=5
```

```
STP=POINT (STF)
```

```
SPP=POINT (SPF)
```

```
TS=1
```

```
RL=NRPLL (STP, SPP, X, Y, TS)
```

(29) PMAX (ポイント:横軸 [0~700])

【機能】

測定ポイント領域 (ポイント1, ポイント2) と、トレースA/Bを指定すると、そのトレース側での領域内の縦軸の最大値点のポイント (横軸: 0~700) を算出します。

【書式】

PMAX (P1, P2, M)

P1 : 指定ポイント1 (0~700)

P2 : 指定ポイント2 (0~700)

M : トレース 0 : トレースA

1 : トレースB

【戻り値】

正常終了時 : 指定ポイント領域内での縦軸の最大値点のポイント (横軸: 0~700)

エラー時 : 不定値

【エラー】

- ・ 指定ポイントが0~700の範囲外の場合、エラーになります。
- ・ エラーのときの戻り値は、不定値になります。
- ・ エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【注意】

P1 > P2 の場合、P1 と P2 を入れ換えて処理します。

【例】

トレースAの0~700ポイントの間のMAX値点のポイント (横軸: 0~700) を求めます。

MP = PMAX (0, 700, 0)

トレースBの周波数10~20MHz間のMAX値点のポイント (横軸: 0~700) を求めます。

F1 = 10E6

P1 = POINT (F1)

MP1 = PMAX (P1, POINT (20E6), 1)

MP2 = PMAX (POINT (F1), POINT (20E6), 1)

(MP1 と MP2 は 同じ)

(30) PMIN (ポイント:横軸 [0~700])

【機能】

測定ポイント領域 (ポイント1, ポイント2) と、トレースA/Bを指定すると、そのトレース側での領域内の縦軸の最小値点のポイント (横軸: 0~700) を算出します。

【書式】

PMIN (P1, P2, M)

P1 : 指定ポイント1 (0~700)

P2 : 指定ポイント2 (0~700)

M : トレース 0 : トレースA

1 : トレースB

【戻り値】

正常終了時 : 指定ポイント領域内での縦軸の最小値点のポイント (横軸: 0~700)

エラー時 : 不定値

【エラー】

- ・ 指定ポイントが0~700の範囲外の場合、エラーになります。
- ・ エラーのときの戻り値は、不定値になります。
- ・ エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【注意】

P1 > P2 の場合、P1 と P2 を入れ換えて処理します。

【例】

トレースAの0~700ポイントの間のMIN値点のポイント (横軸: 0~700) を求めます。

MP = PMIN (0, 700, 0)

トレースBの周波数10~20MHz間のMIN値点のポイント (横軸: 0~700) を求めます。

F1 = 10E6

P1 = POINT (F1)

MP1 = PMIN (POINT (20E6), P1, 1)

MP2 = PMIN (POINT (20E6), POINT (F1), 1)

(MP1 と MP2 は 同じ)

(31) POINT (ポイント:横軸 [0~700])

【機能】

周波数を指定すると、その周波数が測定器内部のポイント(0~700)に対して、何ポイント目にあたるかを算出します。

各ポイント関数を高速に動作させるために重要な関数となります。

【書式】

POINT (F)

F : 指定周波数 (Hz)

【戻り値】

正常終了時 : 周波数を換算したポイント数 (0~700)

エラー時 : 不定値

【エラー】

- ・ 指定周波数がスタート周波数からストップ周波数の範囲外の場合、エラーになります。
- ・ エラーのときの戻り値は、不定値になります。
- ・ エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【例】

周波数3MHzの測定ポイントを求めます。

PO=POINT (3E6)

周波数10MHzの測定ポイントを求めます。

F=10E6

PO=POINT (F)

(32) POWER (総電力)

【機能】

REFレベルが0 dBm時の総電力を求めます。周波数範囲は、ポイント幅 (ポイント1, ポイント2) で決まります。

【書式】

POWER (P1, P2, M)

P1 : 指定ポイント1 (0~700)

P2 : 指定ポイント2 (0~700)

M : トレース 0 : トレースA

1 : トレースB

【戻り値】

正常終了時 : 指定ポイント間P1, P2の総電力 (mW)

エラー時 : 不定値

【エラー】

- ・ 指定ポイント値が0~700の範囲外の場合、エラーになります。
- ・ エラーのときの戻り値は、不定値になります。
- ・ エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【注意】

P1 > P2の場合、P1とP2を入れ換えて処理します。

REFレベルを0 dBm以外に設定し、総電力を求める場合は、REFレベルが10 dBステップごとに設定されたときのみ算出可能となります。

(... -20, -10, 0, 10, 20, ... dBmのみ可能)

算出式を以下に示します。

$$\text{総電力} = \text{POWER}(P1, P2, M) * 10^{\frac{X}{10}} \quad (X \text{ dBm})$$

【例】

トレースAのポイント値 300~400の間の総電力を求めます。(REFレベル=0 dBm時)

$$AW = \text{POWER}(300, 400, 0)$$

トレースAのポイント値 0~700の間の総電力を求めます。(REFレベル=-20 dBm時)

$$AW = \text{POWER}(0, 700, 0) / 100$$

(33) PRPLHN (ポイント:横軸 [0~700])

【機能】

トレースA/B および 周波数軸に対して、左からN番目の極大点のNo. を指定すると、その極大点のポイント(0~700)を算出します。

【書式】

PRPLHN (N, M)

N: 周波数軸に対して左からN番目の極大点のNo.

M: レース 0: トレースA

1: トレースB

【戻り値】

正常終了時: 極大点を周波数軸(横軸:左側から)に対して、N番目の極大点のポイント位置(0~700)

エラー時: 不定値

【エラー】

- ・ N番目の極大点がない場合は、エラーになります。
この関数を使う前に、(27)NRPLHを実行して下さい。
- ・ エラーのときの戻り値は、不定値になります。
- ・ エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【注意】

この関数を使う前には、必ず(27)NRPLH関数を実行して下さい。

【例】

トレースAの0~700ポイント内に対して、微分係数 $Dx=5$, $Dy=3$ の全極大を求め、左から3番目の極大点のポイント位置を求めます。

RH=NRPLH(0, 700, 5, 3, 0)

P=PRPLHN(3, 0)

トレースBの周波数10~20MHz内に対して、微分係数 $Dx=3$, $Dy=5$ の全極大を求め、左から2番目の極大点のポイント位置を求めます。

STF=10E6

SPF=20E6

X=3

Y=5

STP=POINT(STF)

SPP=POINT(SPF)

TS=1

RH=NRPLH(STP, SPP, X, Y, TS)

P=PRPLHN(2, TS)

(34) PRPLL (ポイント: 横軸 [0~700])

【機能】

トレースA/B および 周波数軸に対して、左からN番目の極小点のNo. を指定すると、その極小点のポイント(0~700)を算出します。

【書式】

PRPLL (N, M)

N: 周波数軸に対して左からN番目の極小点のNo.

M: トレース 0: トレースA

1: トレースB

【戻り値】

正常終了時: 極小点を周波数軸(横軸: 左側から)に対して、N番目の極小点のポイント位置(0~700)

エラー時: 不定値

【エラー】

- ・ N番目の極小点がない場合は、エラーになります。
この関数を使う前に、(28) NRPLL を実行して下さい。
- ・ エラーのときの戻り値は、不定値になります。
- ・ エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【注意】

この関数を使う前には、必ず(28) NRPLL関数を実行して下さい。

【例】

トレースAの0~700ポイント内に対して、微分係数 $Dx=5$, $Dy=3$ の全極小を求め、左から3番目の極小点のポイント位置を求めます。

RH=NRPLL (0, 700, 5, 3, 0)

P=PRPLL (3, 0)

トレースBの周波数10~20MHz内に対して、微分係数 $Dx=3$, $Dy=5$ の全極小を求め、左から2番目の極小点のポイント位置を求めます。

STF=10E6

SPF=20E6

X=3

Y=5

STP=POINT (STF)

SPP=POINT (SPF)

TS=1

RH=NRPLL (STP, SPP, X, Y, TS)

P=PRPLL (2, TS)

(35) RPL1 (レベル)

【機能】

測定ポイント領域(ポイント1, ポイント2)と、トレースA/B、微分係数を(Dx, Dy)を指定すると、その指定トレース側のポイント領域内において、極大値、極小値をサーチし、極大値のうちの最大値と、極小値のうちの最小値のレベル差を算出します。

【書式】

RPL1 (P1, P2, Dx, Dy, M)

P1 : 指定ポイント1 (0~700)

P2 : 指定ポイント2 (0~700)

Dx : 微分係数ポイント (0~700)

Dy : 微分係数トレース・データ (0~400)

M : トレース 0 : トレースA

1 : トレースB

【戻り値】

正常終了時 : 極大値のうちの最大値と、極小値のうちの最小値のレベル差(単位は、リファレンス・レベルの単位と同じ)

エラー時 : 不定値

【エラー】

- 指定ポイントが0~700の範囲外の場合、エラーになります。
- 極大値および極小値がサーチできない場合は、エラーになります。
Dx、Dyの数値を変更して下さい。
- エラーのときの戻り値は、不定値になります。
- エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【注意】

P1 > P2の場合、P1とP2を入れ換えて処理します。

【例】

トレースAの0~700ポイント内に対して、微分係数Dx=5, Dy=3の極大値のうちの最大値と、極小値のうちの最小値のレベル差を求めます。

RP=RPL1(0, 700, 5, 3, 0)

トレースBの周波数10~20MHz内に対して、微分係数 Dx=3, Dy=5の極大値のうちの最大値と、極小値のうちの最小値のレベル差を求めます。

STF=10E6

SPF=20E6

X=3

Y=5

STP=POINT(STF)

SPP=POINT(SPF)

TS=1

RP=RPL1(STP, SPP, X, Y, TS)

(36) RTRACE (トレース・データ: [0~400])

【機能】

指定したポイントのトレース・データを返します。

【書式】

RTRACE (P, M)

P: ポイント (0~700)

M: トレース 0: トレースA

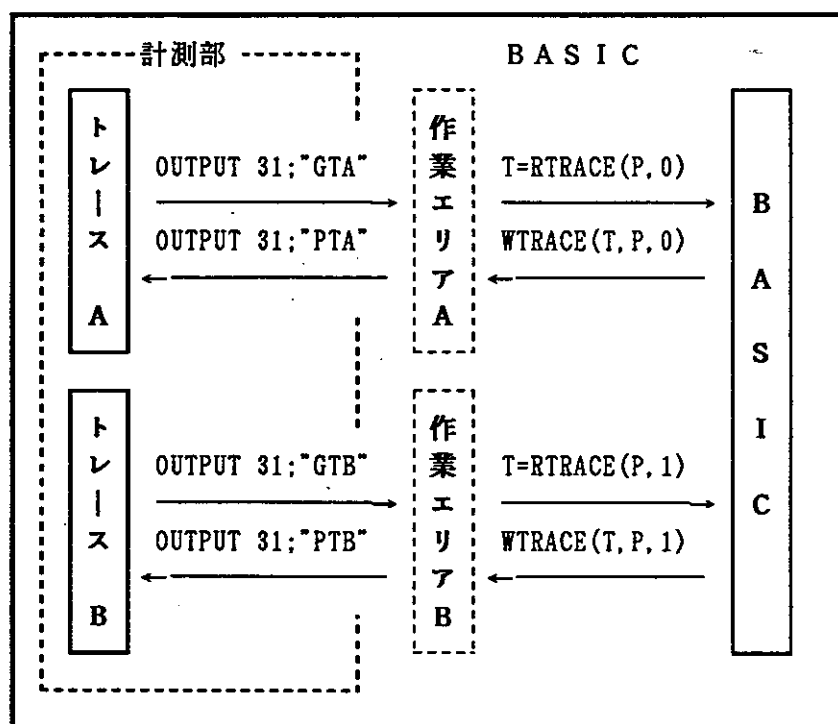
1: トレースB

この命令を行なう前に、必ずOUTPUT 31命令で" GTA"または" GTB"を行ないます。これは計測部にあるトレース・データを作業エリアに転送するために行ないます。

トレースAデータを作業エリアAに転送するときは、OUTPUT 31;" GTA"

トレースBデータを作業エリアBに転送するときは、OUTPUT 31;" GTB"を実行すると、701ポイントすべてを転送します。

このRTRACE関数は、作業エリアからのデータを1ポイントずつ読み込むための関数です。



トレース・データとBASICのデータ転送の関係

【戻り値】

正常終了時: トレース・データ (0~400)

エラー時: 不定値

【エラー】

- ・ ポイント指定で0~700以外の数値は、エラーにならずに不定値が返ります。

【例】

トレースAデータ（0～700ポイント）を配列変数Aに格納します。

```
INTEGER I,A(701)
OUTPUT 31;"GTA"
FOR I=0 TO 700
  A(I+1)=RTRACE(I,0)
NEXT I
```

【参考】

上記のプログラム例の実行速度は、約2.5秒です。
データ1ポイントあたり約3.5ミリ秒です。

(37) VALUE (レベル)

【機能】

ポイント値とトレース (A/B) を指定すると、その指定トレース側での指定ポイントのレベルを求めます。

【書式】

VALUE (P, M)

P : 指定ポイント値 (0~700)

M : トレース 0 : トレースA
1 : トレースB

【戻り値】

正常終了時 : 指定ポイント値のレベル (単位は、リファレンス・レベルの単位と同じ)
エラー時 : 不定値

【エラー】

- ・ トレース・メモリが、A/Bトレース以外の値、または指定ポイントが(0~700)以外の場合は、エラーになります。
- ・ エラーのときの戻り値は、不定値になります。
- ・ エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【例】

トレースAの300ポイント目のレベルを求めます。

L = VALUE (300, 0)

トレースBの周波数3MHzのレベルを求めます。

F = 3 E 6	=	L = CVALUE (3 E 6, 1)
P = POINT (F)		
L = VALUE (P, 1)		

(左右のプログラムは、同じことを求めています。)

【参考】

(7) CVALUE

(31) POINT

(38) VRPLHN (レベル)

【機能】

トレースA/Bおよび周波数軸に対して左からN番目の極大点のNo. を指定すると、その極大点のレベルを算出します。

【書式】

VRPLHN (N, M)

N : 周波数軸に対して左からN番目の極大点のNo.

M : トレース 0 : トレースA

1 : トレースB

【戻り値】

正常終了時 : 極大点を周波数軸 (横軸 : 左側から) に対して、N番目の極大点のレベル (単位は、リファレンス・レベルの単位と同じ)

エラー時 : 不定値

【エラー】

- ・ N番目の極大点がない場合は、エラーになります。
この関数を使う前に、(27) NRPLH を実行して下さい。
- ・ エラーのときの戻り値は、不定値になります。
- ・ エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【注意】

この関数を使う前には、必ず(27) NRPLH関数を実行して下さい。

【例】

トレースAの0~700ポイント内に対して、微分係数 $Dx=5$ 、 $Dy=3$ の全極大を求め、左から3番目の極大点のレベルを求めます。

```
RH=NRPLH (0, 700, 5, 3, 0)
L=VRPLHN (3, 0)
```

トレースBの周波数10~20MHz内に対して、微分係数 $Dx=3$ 、 $Dy=5$ の全極大を求め、左から2番目の極大点のレベルを求めます。

```
STF=10E6
SPF=20E6
X=3
Y=5
STP=POINT (STF)
SPP=POINT (SPF)
TS=1
RH=NRPLH (STP, SPP, X, Y, TS)
L=VRPLHN (2, TS)
```

(39) VRPLLN (レベル)

【機能】

トレースA/Bおよび周波数軸に対して、左からN番目の極小点のNo. を指定すると、その極小点のレベルを算出します。

【書式】

VRPLLN (N, M)

M: トレース 0: トレースA
1: トレースB

N: 周波数軸に対して左からN番目の極小点のNo.

【戻り値】

正常終了時: 極小点を周波数軸(横軸: 左側から)に対して、N番目の極小点のレベル(単位は、リファレンス・レベルの単位と同じ)

エラー時: 不定値

【エラー】

- ・ N番目の極小点がない場合は、エラーになります。
この関数を使う前に、(28) NRPLL を実行して下さい。
- ・ エラーのときの戻り値は、不定値になります。
- ・ エラーのときは、メッセージが画面に出力するだけで、プログラムの実行は停止しません。

【注意】

この関数を使う前には、必ず(28) NRPLL関数を実行して下さい。

【例】

トレースAの0~700ポイント内に対して、微分係数 $Dx=5$ 、 $Dy=3$ の全極小を求め、左から3番目の極小点のレベルを求めます。

```
RH=NRPLL (0, 700, 5, 3, 0)
L=VRPLLN (3, 0)
```

トレースBの10~20MHz内に対して、微分係数 $Dx=3$ 、 $Dy=5$ の全極小を求め、左から2番目の極小点のレベルを求めます。

```
STF=10E6
SPF=20E6
X=3
Y=5
STP=POINT (STF)
SPP=POINT (SPF)
TS=1
RH=NRPLL (STP, SPP, X, Y, TS)
L=VRPLLN (2, TS)
```

(40) WTRACE

【機能】

指定したポイントへトレース・データを書き込みます。

【書式】

WTRACE (T, P, M)

T:トレース・データ (0~400)

P:ポイント (0~700)

M:トレース 0:トレースA

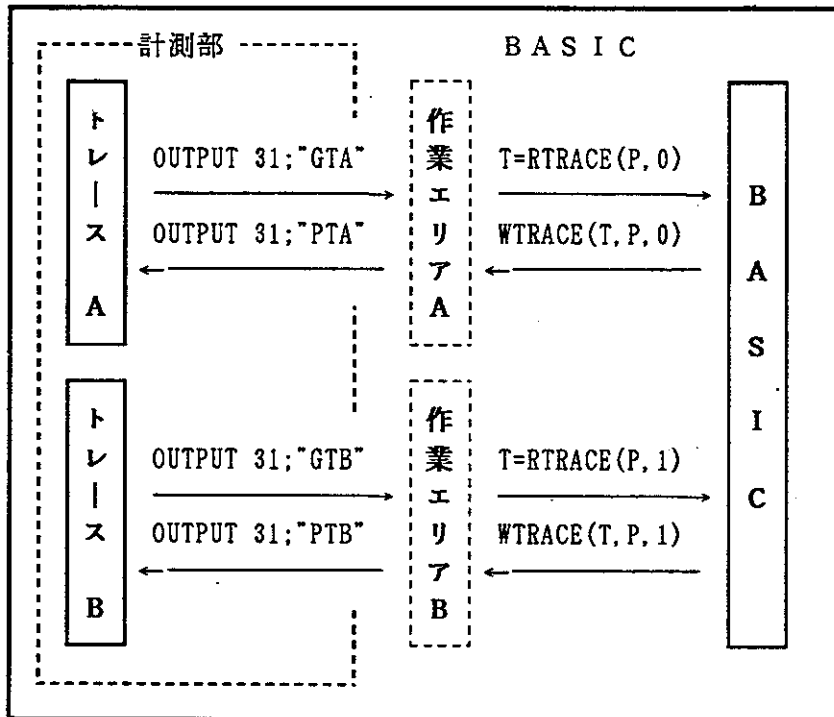
1:トレースB

この命令を実行した後に、必ずOUTPUT 31命令で"PTA"または"PTB"を行ないます。これは作業エリアにあるトレース・データを計測部に転送するために行ないます。

作業エリアAのデータをトレースAへ書き込むときは、OUTPUT 31;"GTA"

作業エリアBのデータをトレースBへ書き込むときは、OUTPUT 31;"GTB"を実行すると、701ポイントすべてを転送します。

このWTRACE関数は、BASICのデータを作業エリアに1ポイントずつ書き込むための関数です。



トレース・データとBASICのデータ転送の関係

【エラー】

- ・ ポイント指定で0~700以外の数値は、エラーにはならず、そしてデータは書き込まれません

【例】

配列変数 B のデータをトレース B データ（計測部）へ転送します。
（このプログラムでは、配列変数 B は 0 です。）

```
INTEGER I, B(701)
FOR I=0 TO 700
  WTRACE(B(I+1), I, 1)
NEXT I
OUTPUT 31: "PTB"
```

【参考】

上記のプログラム例の実行速度は、約 2.2 秒です。
データ 1 ポイントあたり約 8.1 ミリ秒です。

3.4 各種グラフィックの説明

以下の順（アルファベット順）に説明します。

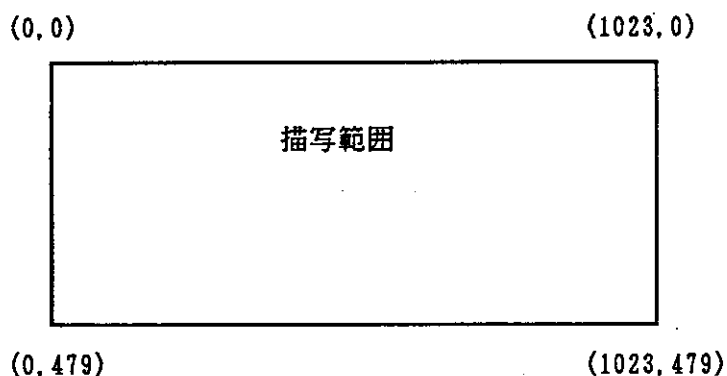
- (1) GADRS
- (2) GFLRECT
- (3) GLINE
- (4) GMKR
- (5) GPOINT
- (6) GRECT
- (7) GSTK
- (8) GSTYLE

(1) GADRS

【機能】

描写画面のアドレス・モードを指定します。
アドレス・モードには以下の2つがあります。

- 0 : 絶対アドレス
- 1 : ビューポート・アドレス



絶対アドレス・モードを指定すると原点は左上となり、このときのX, Y指定は無視されます。また、ビューポート・アドレスを指定したときのX, Yが原点となります。

注) X, Yは絶対アドレスで指定し、上図の範囲を超えないようにして下さい。

ビューポート・アドレスを指定したときは、原点を基準として右上が第1象限になります。

【書式】

GADRS (Mo, X, Y)

Mo : モード 0 : 絶対アドレス
 1 : ビューポート・アドレス

X : 横軸 (0~1023)

Y : 縦軸 (0~479)

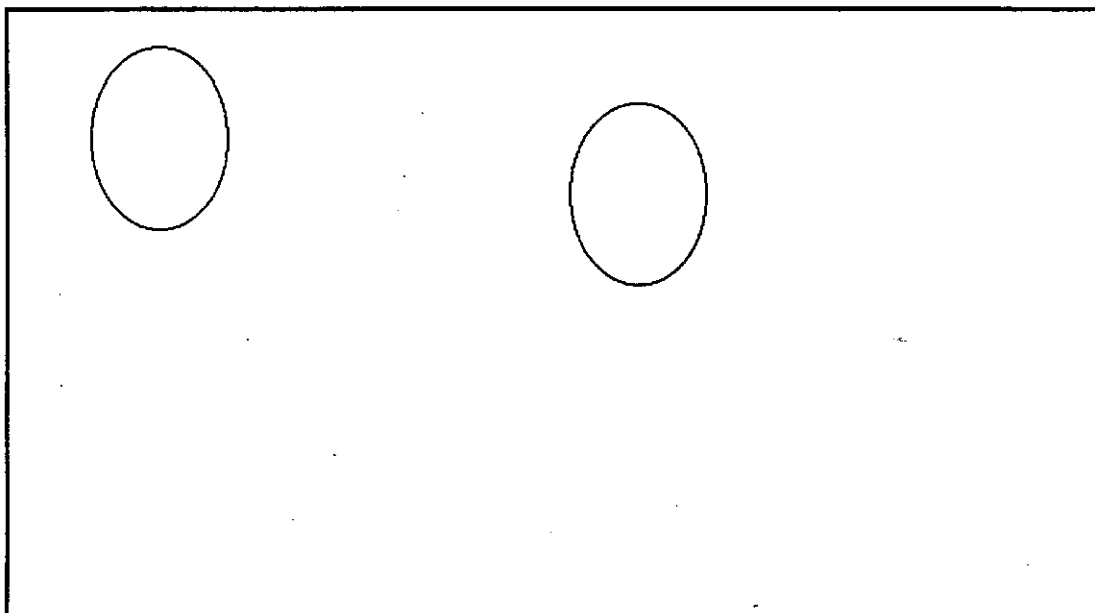
【エラー】

- ・ ビューポート・アドレス指定のときにX, Yが、描写画面の範囲外に設定されると、エラーになります。
- ・ エラー時は、メッセージを出力するだけで、実行は停止しません。

【例】

```
OUTPUT 31;"VS3"  
CLS 1  
R=100  
FOR M=0 TO 1  
  GADRS(M, 512, 240)  
  FOR I=0 TO R*2 STEP R/90  
    X=SIN(I)*R  
    Y=COS(I)*R  
    GPOINT(1, X, Y)  
  NEXT I  
NEXT M
```

【実行結果】



(2) GFLRECT

【機能】

指定された2点間(座標1、座標2)を対角線とする長方形を塗りつぶして描きます。

【書式】

```
GFLRECT (D, X1, Y1, X2, Y2)
```

D : 0 : 消去
1 : 描く

X1, Y1 : 座標1

X2, Y2 : 座標2

【エラー】

- ・ X, Yが、描写画面の範囲外に設定されると、エラーになります。(ビューポート・アドレス時は注意して下さい。)
- ・ エラー時は、メッセージを出力するだけで実行は停止しません。

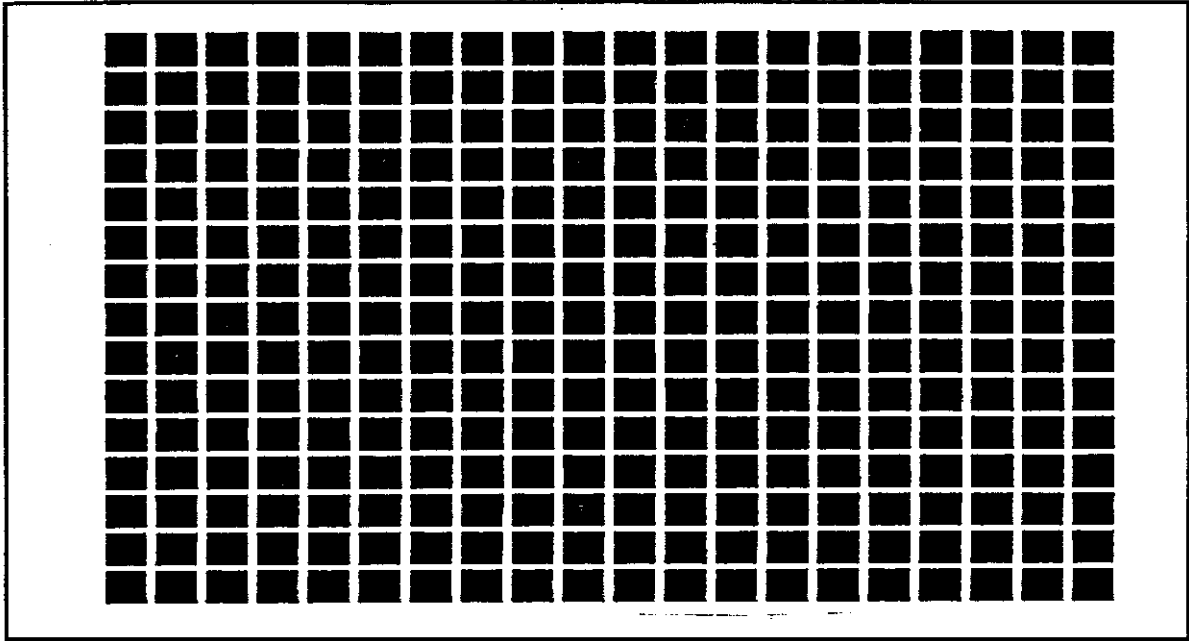
【注意】

- ・ 座標1は必ず左上、座標2は右下となるように指定して下さい。

【例】

```
OUTPUT 31;"VS3"  
CLS 1  
INTEGER X,Y  
FOR X=10 TO 970 STEP 50  
  FOR Y=10 TO 450 STEP 30  
    GFLRECT(1,X,Y,X+40,Y+25)  
  NEXT Y  
NEXT X  
STOP
```

【実行結果】



(3) G LINE

【機能】

指定された2点間(座標1、座標2)に線を描きます。

【書式】

GLINE (S, D, X1, Y1, X2, Y2)

S : スタイル 0 : 実線
1 : 破線
2 : 点線
3 : 一点鎖線

D : 0 : 消去
1 : 描く

X1, Y1 : 座標1

X2, Y2 : 座標2

【エラー】

- ・ X, Yが、描写画面の範囲外に設定されると、エラーになります。(ビューポート・アドレス時は注意して下さい。)
- ・ エラー時は、メッセージを出力するだけで、実行は停止しません。

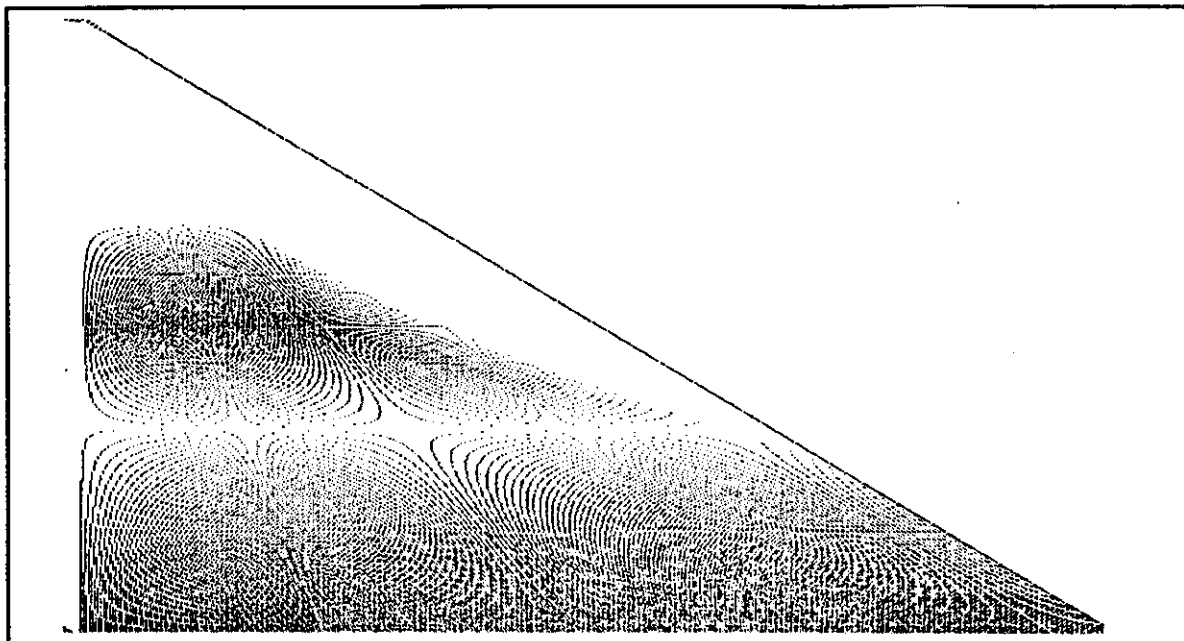
【注意】

- ・ 実線を書く以外は、ライン・スタイル((8) のGSTYLE関数) であらかじめスタイルを指定しておきます。

【例】

```
OUTPUT 31;"VS3"  
CLS 1  
FOR I=2 TO 1023 STEP 3  
  GLINE(0,1,2,0,I,479)  
  GLINE(0,1,0,0,I-2,479)  
NEXT I  
STOP
```

【実行結果】



(4) GMKR

【機能】

指定した位置にマーカ（ノーマル、 Δ ）を描きます。

【書式】

GMKR (MK, D, X, Y)

MK : マーカ種類 0 : ノーマル・マーカ
1 : Δ マーカ

D : 0 : 消去
1 : 描く

X, Y : 座標

【エラー】

- ・ X, Yが、描写画面の範囲外に設定されると、エラーになります。（ビューポート・アドレス時は注意して下さい。）
- ・ エラー時は、メッセージを出力するだけで、実行は停止しません。

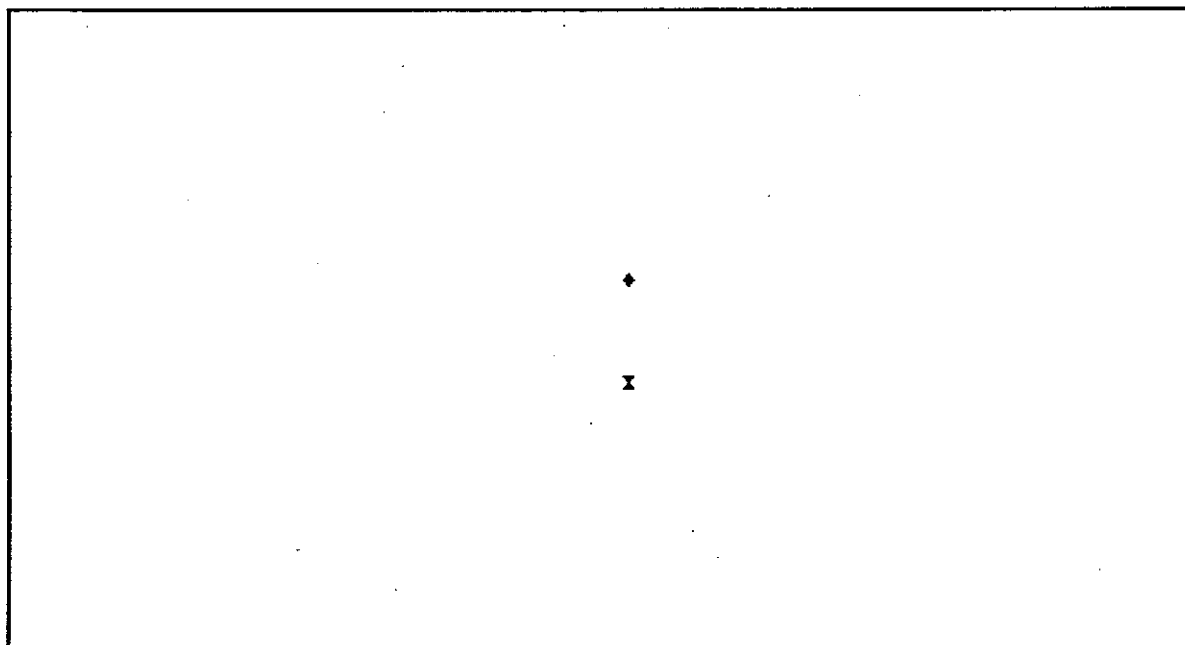
【注意】

- ・ マーカのサイズは、11×11ドットです。
- ・ マーカの中心が指定した位置に描かれます。

【例】

```
OUTPUT 31;"VS3"  
CLS 1  
GMKR(0,1,512,200)  
GMKR(1,1,512,280)
```

【実行結果】



(5) GPOINT

【機能】

指定した位置に点を描きます。

【書式】

GPOINT (D, X, Y)

D : 0 : 消去
 1 : 描く
X, Y : 座標

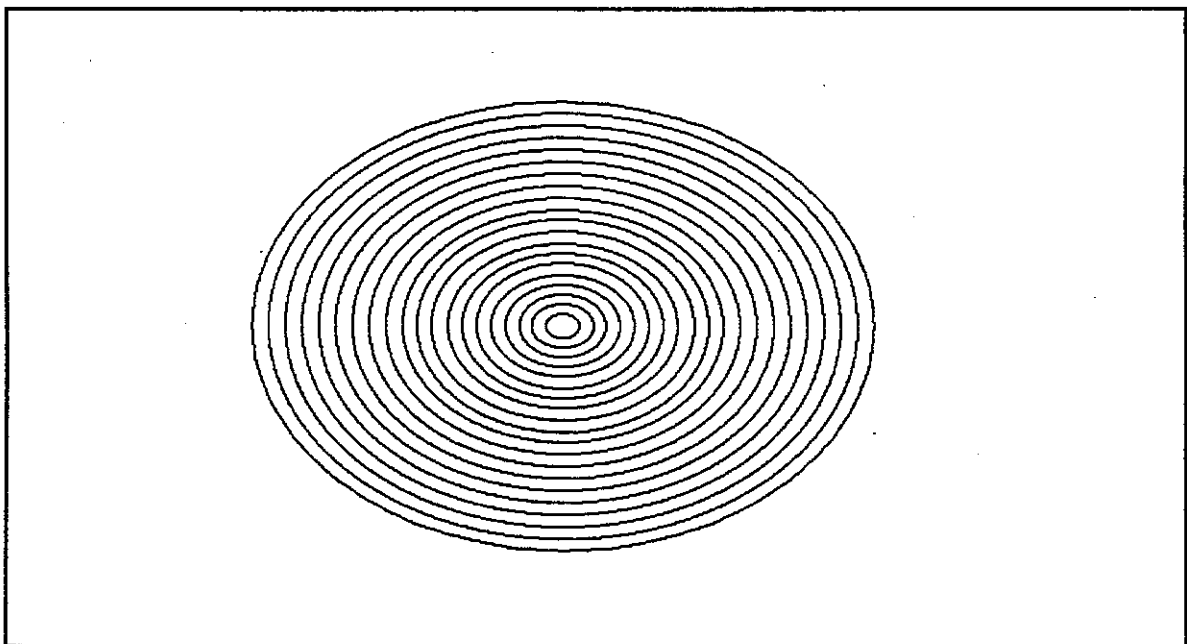
【エラー】

- ・ X, Yが、描写画面の範囲外に設定されると、エラーになります。(ビューポート・アドレス時は注意して下さい。)
- ・ エラー時は、メッセージを出力するだけで、実行は停止しません。

【例】

```
OUTPUT 31;"VS3"  
CLS 1  
FOR R=200 TO 1 STEP -10  
  FOR I=0 TO R*2 STEP R/270  
    X=SIN(I)*R*1.5+512  
    Y=COS(I)*R+240  
    GPOINT(1,X,Y)  
  NEXT I  
NEXT R
```

【実行結果】



(6) GRECT

【機能】

指定された2点間(座標1、座標2)を対角線とする長方形を描きます。

【書式】

GRECT (S, D, X1, Y1, X2, Y2)

S :スタイル 0:実線
1:破線
2:点線
3:一点鎖線

D :0:消去
1:描く

X1, Y1:座標1

X2, Y2:座標2

【エラー】

- ・ X, Yが、描写画面の範囲外に設定されるとエラーになります。(ビューポート・アドレス時は注意して下さい。)
- ・ エラー時は、メッセージを出力するだけで、実行は停止しません。

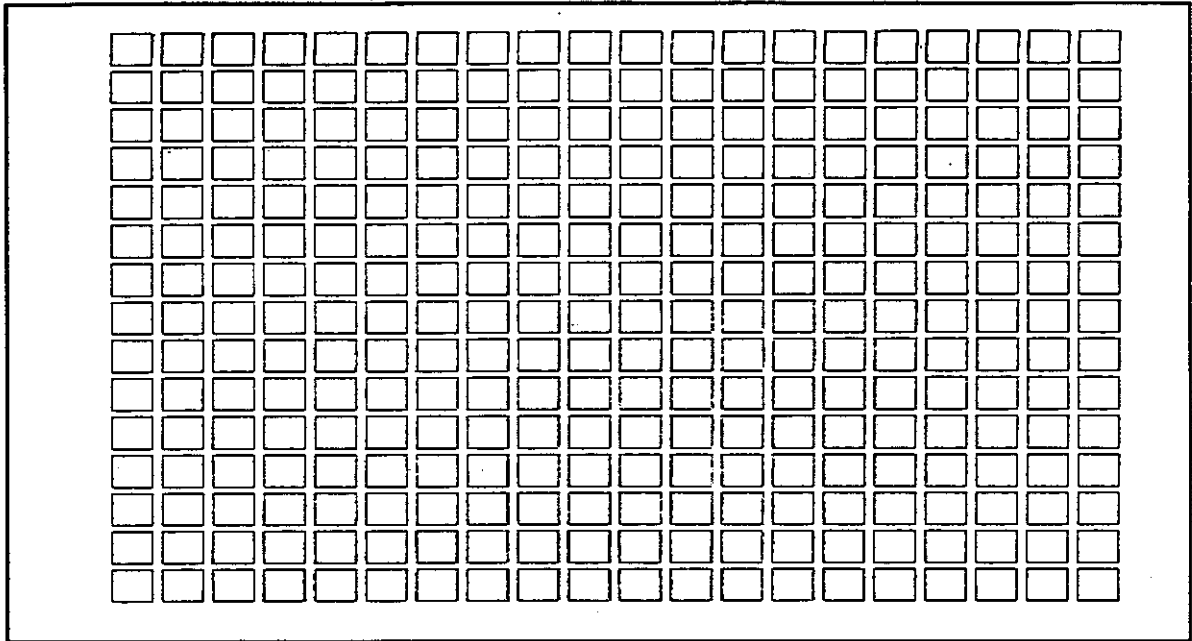
【注意】

- ・ 実線を書く以外は、ライン・スタイル((8)のGSTYLE関数)であらかじめスタイルを指定しておきます。

【例】

```
OUTPUT 31:"VS3"  
CLS 1  
INTEGER X,Y  
FOR X=10 TO 970 STEP 50  
  FOR Y=10 TO 450 STEP 30  
    GRECT(0,1,X,Y,X+40,Y+25)  
  NEXT Y  
NEXT X  
STOP
```

【実行結果】



(7) GSTR

【機能】

グラフィック画面の指定位置に文字を描きます。

【書式】

GSTR (C, X, Y, STR)

C : 文字サイズ 0 : 16×20ドット
1 : 18×24ドット
X : 横軸 (絶対アドレス: 0~1023)
Y : 縦軸 (絶対アドレス: 0~479)
STR : 文字列表現式 (18文字まで)

【エラー】

- ・ X, Yが、描写画面の範囲外に設定されると、エラーになります。(ビューポート・アドレス時は注意して下さい。)
- ・ エラー時は、メッセージを出力するだけで、実行は停止しません。

【注意】

- ・ 文字列は18文字までしか出力されません。19文字以上は無視されます。

【例】

```
OUTPUT 31;"VS3"  
SS="ADVANTEST"  
FOR I=0 TO 450 STEP 25  
  GSTR(I%2, I, I, SS)  
NEXT I  
STOP
```

【実行結果】

```
ADVANTEST  
  ADVANTEST  
    ADVANTEST  
      ADVANTEST  
        ADVANTEST  
          ADVANTEST  
            ADVANTEST  
              ADVANTEST  
                ADVANTEST  
                  ADVANTEST  
                    ADVANTEST  
                      ADVANTEST  
                        ADVANTEST  
                          ADVANTEST  
                            ADVANTEST  
                              ADVANTEST  
                                ADVANTEST
```

(8) G S T Y L E

【機能】

破線、点線および一点鎖線のそれぞれの要素の長さを指定します。

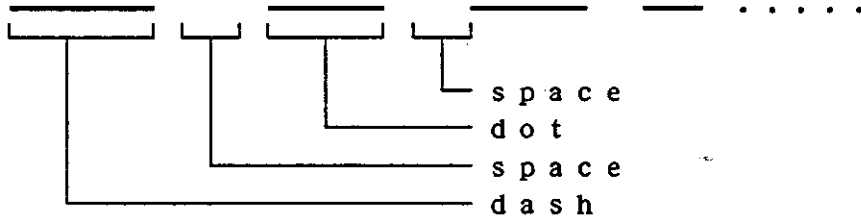
【書式】

G S T Y L E (d a s h , s p a c e , d o t)

この関数は、破線、点線、および一点鎖線のそれぞれの要素の長さを指定するものです。それぞれの要素とは破線部、スペース部、点線部のことを差し、この要素は以下の組合せで構成されています。

破線 : d a s h + s p a c e
点線 : d o t + s p a c e
一点鎖線 : d a s h + s p a c e + d o t + s p a c e

例) 1点鎖線を指定のときは以下ようになります。



【例】

```
OUTPUT 31;"VS3"  
CLS 1  
S=0  
GSTYLE(20,10,5)  
FOR X=1 TO 1000  
  IF X % 350 = 1 THEN ++S : X+=10  
  GLINE(S,1,X,0,X,479)  
NEXT X  
STOP
```


MEMO



索引

【B】

BND -----	3 - 12
BNDH -----	3 - 14
BNDL -----	3 - 13
BUZZER -----	2 - 7

【C】

CALL -----	2 - 9
CAT -----	2 - 11
CBND -----	3 - 15
CBNDH -----	3 - 17
CBNDL -----	3 - 16
CLEAR -----	2 - 13
CLOSE # -----	2 - 16
CLS -----	2 - 14
CONT -----	2 - 18
CONTROL -----	2 - 19
CVALUE -----	3 - 18
CVRSOR (CSRと省略可) -----	2 - 22

【D】

DCVALUE -----	3 - 19
DELIMITER -----	2 - 24
DFREQ -----	3 - 20
DIM -----	2 - 26
DISABLE INTR -----	2 - 28
DLEVEL -----	3 - 21
DPOINT -----	3 - 22
DVALUE -----	3 - 23

【E】

ENABLE INTR -----	2 - 30
ENTER # -----	2 - 31
ENTER (ENTと省略可) -----	2 - 35

【F】

FMAX -----	3 - 24
FMIN -----	3 - 25
FOR.....TO.....STEP BREAK/CONTINUE~NEXT -----	2 - 39
FREQ -----	3 - 26
FRPLHN -----	3 - 27
FRPLLN -----	3 - 28

【G】

GADRS -----	3 - 57
GFLRECT -----	3 - 59
GLINE -----	3 - 61
GLIST -----	2 - 42
GLISTN -----	2 - 44
GMKR -----	3 - 63
GOSUB~RETURN -----	2 - 46
GOTO -----	2 - 48
GPOINT -----	3 - 64
GPRINT -----	2 - 49
GPRINT USING (USEと省略可) -----	2 - 51
GRECT -----	3 - 65
GSTK -----	3 - 67
GSTYLE -----	3 - 68

【I】

IF THEN ELSE/ ELSE IF/END IF -----	2 - 53
INITIALIZE (INITと省略可) -----	2 - 56
INPUT (INPと省略可) -----	2 - 60
INTEGER -----	2 - 57
INTERFACE CLEAR -----	2 - 59

【L】

LEVEL -----	3 - 29
LIST -----	2 - 62
LISTN -----	2 - 63
LLIST -----	2 - 64
LLISTN -----	2 - 65
LMTMD1 -----	3 - 30
LMTMD2 -----	3 - 31
LMTUL1 -----	3 - 32
LMTUL2 -----	3 - 33
LOCAL -----	2 - 67
LOCAL LOCKOUT -----	2 - 68
LPRINT -----	2 - 69
LPRINT USING (USEと省略可) -----	2 - 71
LVDPPOINT -----	3 - 34
LVPOINT -----	3 - 35

【M】

MAX -----	3 - 36
MIN -----	3 - 37

【N】	
NRPLH -----	3 - 38
NRPLL -----	3 - 40

【O】	
OFF END -----	2 - 73
OFF ERROR -----	2 - 74
OFF KEY -----	2 - 75
OFF SRQ/ISRQ -----	2 - 77
ON END ~	
GOTO/GOSUB -----	2 - 78
ON ERROR ~	
GOTO/GOSUB -----	2 - 80
ON KEY ~	
GOTO/GOSUB -----	2 - 82
ON SRQ/ISRQ ~	
GOTO/GOSUB -----	2 - 84
OPEN # -----	2 - 87
OUTPUT # -----	2 - 90
OUTPUT (OUTと省略可) -----	2 - 93

【P】	
PAUSE -----	2 - 96
PMAX -----	3 - 42
PMIN -----	3 - 43
POINT -----	3 - 44
POWER -----	3 - 45
PPRINTF (PRFと省略可) -----	2 - 97
PRINT (?と省略可) -----	2 - 100
PRINT USING (USEと省略可) -----	2 - 102
PRINTER -----	2 - 104
PRPLHN -----	3 - 46
PRPLLN -----	3 - 47
PURGE -----	2 - 113

【R】	
READ DATA/RESTORE -----	2 - 106
REM (!と省略可) -----	2 - 108
REMOTE -----	2 - 109
RENAME -----	2 - 110
REQUEST -----	2 - 111
RPL1 -----	3 - 48
RTRACE -----	3 - 49

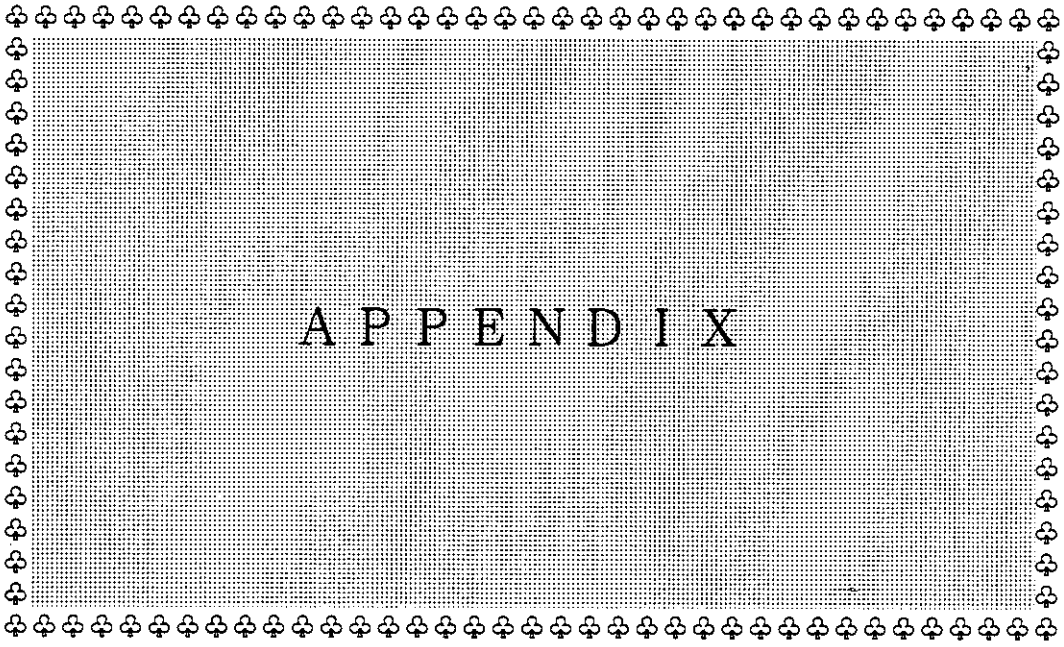
RUN -----	2 - 112
-----------	---------

【S】	
SCRATCH -----	2 - 114
SELECT CASE/	
CASE ELSE/END SELECT --	2 - 115
SEND -----	2 - 117
SPOLL -----	2 - 119
SPRINTF -----	2 - 120
STEP -----	2 - 123
STOP -----	2 - 124

【T】	
TRIGGER -----	2 - 125

【V】	
VALUE -----	3 - 51
VRPLHN -----	3 - 52
VRPLLN -----	3 - 53

【W】	
WAIT -----	2 - 126
WTRCE -----	3 - 54



APPENDIX

付 録 1 の 目 次

A1.1	機能別コマンドとステートメント一覧	A-1
A1.2	機能別ビルトイン関数とグラフィック機能一覧	A-4
A1.3	パラメータ指定一覧	A-6
A1.4	エラー・メッセージ一覧	A-9

機 能	コマンドとステートメント	内 容
①コマンド	CONT CONTROL LIST LLIST LISTN LLISTN RUN SCRATCH STEP	プログラム停止後の再実行を行なう。 各制御に関する値を設定する。 プログラム・リストを表示する。 プログラム・リストを表示する。(RS-232) プログラム・リストを表示する。 プログラム・リストを表示する。(RS-232) プログラムを実行させる。 BASIC内のプログラムを消去する。 プログラムを1行実行する。
②算術関数	ABS ATN COS LOG SIN SQR TAN	与えられた値の絶対値を求める。 与えられた値の逆正接値を求める。 与えられた値の余弦値を求める。 与えられた値の自然対数を求める。 与えられた値の正弦値を求める。 与えられた値の平方根を求める。 与えられた値の正接値を求める。
③ビット演算	BAND BNOT BOR BXOR	ビットANDを求める。 ビットNOTを求める。 ビットORを求める。 ビットXORを求める。
④割り込み 制 御	ENABLE INTR DISABLE INTR ON END ON KEY ON ISRQ ON SRQ ON ERROR OFF END OFF KEY OFF ISRQ OFF SRQ OFF ERROR	割り込み受信許可状態にする。 割り込み受信禁止状態にする。 EOF割り込みの分岐を定義する。 キー割り込みの分岐を定義する。 本体計測部SRQ割り込みの分岐を定義する。 GPIB、SRQ割り込みの分岐を定義する。 エラー発生割り込みの分岐を定義する。 EOF割り込みの分岐定義を解除する。 キー割り込みの分岐定義を解除する。 本体計測部SRQ割り込みの分岐定義を解除する。 GPIB、SRQ割り込みの分岐定義を解除する。 エラー発生割り込みの分岐定義を解除する。
⑤文字列操作	NUM CHRS LEN POS SPRINTF	文字列の先頭の文字のASCIIコードを求める。 数値をASCII文字に変換する。 文字列の長さを求める。 文字列2の中から文字列1がある位置を求める。 書式を決め文字列変数に代入する。

機能	コマンドとステートメント	内容
⑥メモリカード制御	CAT CALL CLOSE # ENTER # OPEN # OUTPUT # INITIALIZE(INIT) PURGE RENAME	メモリ・カードの内容を表示する。 サブ・プログラムをロードする。 ファイルを閉じる。 ファイルからデータを読み込む。 ファイルを開く。 ファイルにデータを書き込む。 メモリ・カードを初期化する。 指定ファイルを消去する。 ファイル名を変更する。
⑦画面制御	CURSOR(CSR) CLS	指定位置にカーソルを移動する。 画面を消去する。
⑧ステートメント	BUZZER DIM FOR TO STEP NEXT BREAK CONTINUE GOSUB RETURN GOTO IF THEN ELSE END IF INPUT(INP) INTEGER LPRINT LPRINT USING(USE) PAUSE PRINT(?) PRINT USING(USE) PRINTER PRINTF(PRF) READ DATA RESTORE REM(!) SELECT CASE END SELECT STOP WAIT	ブザーを鳴らす。 配列変数を宣言する。 繰り返し処理を設定する。 繰り返し処理から抜ける。 繰り返し処理を先頭に戻す。 サブ・ルーチン分岐する。 サブ・ルーチンからの復帰する。 指定位置へ分岐する。 条件判断をして処理する。 変数への入力を行なう。 整数型変数の宣言する。 プリント(RS-232)出力する。 プリント(RS-232)出力する。(書式指定) 一時停止する。 文字を画面に表示する。 文字を画面に表示する。(書式指定) GPIBプリンタ装置のアドレス指定を行なう。 文字を画面に表示する。(書式指定) DATA文からデータを読み取り、変数へ代入する。 READ文で読むDATA文を指定する。 注釈文 条件判断を行い処理する。 プログラム実行停止する。 指定時間、実行停止する。

機 能	コマンドとステートメント	内 容
⑨GPIB関係 コマンド	CLEAR DELIMITER ENTER(ENT) GLIST GLISTN GPRINT GPRINT USING(USE) INTERFACE CLEAR LOCAL LOCAL LOCKOUT OUTPUT(OUT) REMOTE REQUEST SEND SPOLL TRIGGER	DCL、SDCを送出する。 デリミタを設定する。 GPIBデータを入力する。(パラレルI/O含む) GPIBプリンタへプログラム・リストを出力する。 GPIBプリンタへプログラム・リストを出力する。 データをGPIBプリンタへ出力する。 データをGPIBプリンタへ出力する。 (書式指定) IFCを送出する。 指定装置をローカル状態にする。 指定装置をローカル・ロックアウト状態にする。 データをGPIBに出力する。 (パラレルI/O含む) 指定装置をリモート状態にする。 標準GPIBにSRQを出力する。 GPIBデータを1つずつ出力する。 指定装置のシリアル・ポールを行なう。 GETを送出する。

機 能	関 数	内 容
①周波数を求める	FREQ DFREQ FMAX FMIN BND BNDL BNDH CBND CBNDL CBNDH FRPLHN FRPLL	ポイント値から周波数を求める ポイント間の周波数を求める ポイント値で指定された2点間の最大レベル位置の周波数を求める ポイント値で指定された2点間の最小レベル位置の周波数を求める ポイント値で指定された位置でのLOSSレベルの帯域幅を求める ポイント値で指定された位置でのLOSSレベル位置での低周波数側を求める ポイント値で指定された位置でのLOSSレベル位置での高周波数側を求める 周波数で指定された位置でのLOSSレベルの帯域幅を求める 周波数で指定された位置でのLOSSレベル位置での低周波数側を求める 周波数で指定された位置でのLOSSレベル位置での高周波数側を求める 周波数軸に対して左からN番目の極大点の周波数を求める 周波数軸に対して左からN番目の極小点の周波数を求める
②ポイントを求める (周波数ポイント：横軸)	POINT DPOINT PMAX PMIN PRPLHN PRPLL	周波数のポイント(横軸)を求める 指定周波数間のポイント数を求める ポイント値で指定された2点間の最大レベル位置の周波数ポイント(横軸)を求める ポイント値で指定された2点間の最小レベル位置の周波数ポイント(横軸)を求める 周波数軸に対して左からN番目の極大点のポイントを求める 周波数軸に対して左からN番目の極小点のポイントを求める
③ポイントを求める (トレース・データ縦軸)	LVPOINT LVDPOINT	レベルからトレース・データのポイント(縦軸)値を求める レベル間のトレース・データのポイント(縦軸)値を求める

機 能	関 数	内 容
④レベルを求める	VALUE	ポイント値で与えられた周波数位置のレベルを求める
	DVALUE	ポイント値で与えられた2点の周波数位置のレベル差を求める
	CVALUE	周波数位置のレベルを求める
	DCVALUE	与えられた2点の周波数位置のレベル差を求める
	LEVEL	ポイント(縦軸)値で与えられたトレース・データのレベルを求める
	DLEVEL	ポイント(縦軸)間でレベルを求める
	MAX	ポイント値で指定された2点間の最大レベルを求める
	MIN	ポイント値で指定された2点間の最小レベルを求める
	RPL1	極大点の最大値と極小点の最小値のレベル差を求める
VRPLHN	周波数軸に対して左からN番目の極大点のレベルを求める	
VRPLLN	周波数軸に対して左からN番目の極小点のレベルを求める	
⑤総電力を求める	POWER	ポイント値で指定された2点間の総電力を求める
⑥リップル (極大, 極小) の数を求めます。	NRPLH	すべての極大を周波数軸に対して求め, その極大点の数を求める
	NRPLL	すべての極小を周波数軸に対して求め, その極小点の数を求める
⑦上下限チェックをする	LMTMD1	基準値, 基準値幅, 指定による上下限領域チェックを行なう
	LMTMD2	ポイント値で指定された周波数位置でのレベルを基準値, 基準値幅, 指定により上下限領域チェックを行なう
	LMTUL1	上下限領域チェックを行なう
	LMTUL2	ポイント値で指定された周波数位置でのレベルの上下限領域チェックを行なう
⑧トレース・データ (read/write)機能	RTRACE	指定ポイントのトレース・データ読み込む
	WTRACE	指定ポイントのトレース・データ書き込む
⑨グラフィック機能	GADRS	グラフィック・ポイントの絶対アドレス/ビューポートアドレスを指定する
	GSTR	文字列を描く
	GLINE	指定2点間に直線を描く
	GSTYLE	破線, 点線, 1点鎖線の要素の長さを指定する
	GPOINT	指定した位置に点を描く
	GRECT	指定2点間を対角線とする長方形を描く
	GFLRECT	指定2点間を対角線とする長方形を塗りつぶす
	GMKR	指定した位置にマーカ(ノーマル/デルタ)を描く

```
F = FREQ (P)
F = DFREQ (P1, P2)
F = FMAX (P1, P2, M)
F = FMIN (P1, P2, M)
F = BND (P, X, M)
F1 = BNDL (P, X, M)
Fh = BNDH (P, X, M)
F = CBND (F, X, M)
F1 = CBNDL (F, X, M)
Fh = CBNDH (F, X, M)
F = FRPLHN (N, M)
F = FRPLLN (N, M)

P = POINT (F)
P = DPOINT (F1, F2)
P = PMAX (P1, P2, M)
P = PMIN (P1, P2, M)
P = PRPLHN (N, M)
P = PRPLLN (N, M)

T = LVPOINT (L)
T = LVDPOINT (L1, L2)

L = VALUE (P, M)
L = DVALUE (P1, P2, M)
L = CVALUE (F1, F2, M)
L = DCVALUE (F1, F2, M)
L = LEVEL (T)
L = DLEVEL (T1, T2)
L = MAX (P1, P2, M)
L = MIN (P1, P2, M)
L = RPL1 (P1, P2, Dx, Dy, M)
L = VRPLHN (N, M)
L = VRPLLN (N, M)

W = POWER (P1, P2, M)

N = NRPLH (P1, P2, Dx, Dy, M)
N = NRPLL (P1, P2, Dx, Dy, M)

C = LMTMD1 (Dd, S, Ds)
C = LMTMD2 (P, S, Ds, M)
C = LMTUL1 (Dd, Up, Lo)
C = LMTUL2 (P, Up, Lo, M)

T = RTRACE (P, M)
WTRACE (T, P, M)
```

注) この関数は値を返しません。

F : 周波数
P : ポイント (0 ~ 700)
L : レベル
T : トレース・データ (0 ~ 400)
M : トレースA/B (0/1)
X : lossレベル
C : チェック値 (0, 1, 2, -1)
N : リップル数
W : ワット (電力)

Dx : 横軸微分係数
Dy : 縦軸微分係数
S : 基準値
Ds : 基準値幅
Dd : 被検データ
Lo : 下限値
Up : 上限値

【グラフィック】

```
GADRS (Mo, X, Y)
GFLRECT (D, X1, Y1, X2, Y2)
GLINE (S, D, X1, Y1, X2, Y2)
GMKR (MK, D, X, Y)
GPOINT (D, X, Y)
GRECT (S, D, X1, Y1, X2, Y2)
GSTR (C, X, Y, STR)
GSTYLE (dash, space, dot)
```

C	: 文字サイズ	0	; 16×20ドット
		1	; 18×24ドット
D	: セット/消去	0	; 消去
		1	; セット (描く)
MK	: マーカ	0	; ノーマル・マーカ
		1	; △マーカ
Mo	: アドレス・モード	0	; 絶対アドレス
		1	; ビューポート・アドレス
S	: ライン・スタイル	0	; 実線
		1	; 破線
		2	; 点線
		3	; 一点鎖線
STR	: 文字列表現式		
X	: 座標 (横軸)		
Y	: 座標 (縦軸)		
dash	: 破線部		
dot	: 点線部		
space	: スペース部		

A1.4 エラーメッセージ一覧

記述上の注意 xxx : 文字列を表わします
yy : 数値を表わします

(1) ateエディタのエラー・メッセージ一覧


エラーメッセージ	説明
Already auto line no. mode	すでに自動行番号挿入モードであるにもかかわらず、再度自動行番号挿入モードにしようとした
Cannot allocate バイト数 bytes	エディタ内部に確保できるメモリがない
Cannot allocate memory	エディタ内部に確保できるメモリがないため、リナンバリングできない
Cannot allocate WINDOW block	エディタ内部に確保できるメモリがないため、新しいウィンドウをオープンできない
Cannot create buffer	エディタ内部に確保できるメモリがない
Cannot find error line	BASIC実行時のエラー行が、ateエディタ内プログラムにない
Cannot open file for writing	メモ리카ードがない、WRITE PROTECTがONなどにより、セーブするファイルがオープンできない
Cannot split a ライン数 line window	ウィンドウの行数が少ないために、スプリットできない
Line no. is out of range	行数が65535を超えている
No file name	メモ리카ードにセーブするとき、ファイル名がない
No mark set in this window	削除、コピーなど範囲指定のマークがない
Not found	検索する文字列がない
Not line no. mode	リナンバリングを実行したとき、自動行番号挿入モードでなかった
Too large region	削除、コピーなど範囲指定の範囲が大きすぎる
Write I/O error	メモ리카ードがない、電池切れ等により、メモ리카ードをアクセスできない

エラー・メッセージ	内 容
yy error(s) appeared	ラベル、行ナンバのエラー
"xxx" file cannot be opened.	オープンしようとするファイルがない
"xxx" file is already opened with another PATH.	すでにオープンしてあるファイルをオープンしようとした
"xxx" file is already exist.	オープン時のモードと使用した命令が違う
"xxx" read error.	読み込みエラー
0 divide	0による除算 ($n/0$) が行なわれた
Array's range error	配列変数の添字が宣言よりも大きい
Bad free call	メモリ管理上のエラー
CANNOT assigned into this token	文字列変数に代入できない
cannot read data from "xxx" file.	読み込むべきファイルがない
cannot specify "USING"	読み込みバイト数と読み込んだバイト数が違う
cannot write data into "xxx" file.	書き込むべきファイルがない
end of "xxx" file	EOF (End Of File) まで読み込んだ
expression format error	書式の表現が誤っている
file format error	256文字以内にターミネータがない
file is NOT open.	オープンしないで書き込み/読み込みを実行した
FOR <init value> does NOT exist	FOR文の初期値がない
FOR variable does NOT exist.	FOR文のカウンタ変数がない
FOR's nest is abnormal.	FOR文が入れ子 (nest) できない
Invalid dimension parameter	配列変数のパラメータが誤っている
Invalid string constant	ダブル・クォーテーションがあわない
invalid type in xxx	xxxの中に無効なものがある

エラー・メッセージ	内 容
label not found	指定したラベルがない
Label xxx is already exists	x x x のラベルはすでに存在している
Line No.yy is out of range.	行番号の指定が範囲を超えている
memory space full	メモリに空き領域がない
NO operand in xxx	x x x の演算書式が間違っている
NOT available ASCII char(xx)	アスキー・コードが有効でない
Not found DATA statement	R E S T O R Eする先にDATA文がない
Not found THEN in xxx	I F文の後ろにT H E Nがない
Only one INPUT file can be opened.	読み込みモードで2つ以上オープンしようとした
Only one OUTPUT file can be opened.	書き込みモードで2つ以上オープンしようとした
Overflow value	数値が扱える範囲を超えている
parameter error	パラメータが誤っている
Program CANNOT be continued.	終了したプログラムを再実行しようとした
Program NOT exist	プログラムが存在しないのに実行した
SELECT nesting overflow	S E L E C T文の入れ子が多すぎる
string declaration error	ダブル・クォーテーションがあわない
string length is too long	文字列変数の宣言が多すぎる(最大128)
Substring error	サブ・ストリングの指定が誤っている
Unbalanced BREAK	B R E A K文がF O R ~ N E X T間がない
Unbalanced FOR variable in NEXT	F O R文とN E X T文の関係がおかしい
Unbalanced line No.	L I S Tで指定した行がない
Unbalanced NEXT statement	F O R文があるのにN E X T文がない

エラー・メッセージ	内 容
Unbalanced xxx	構文上、バランス(カッコ、カギカッコ等)がとれない
Unbalanced xxx block	xxxのブロック (FOR, IF文等) があわない
Undefined LABEL	ラベルが存在しない
undefined ON condition	ON条件が未定義の状態、ON状態が発生した
Uninstalled type (xxx)	変数の書式が間違っている
Unknown line No.	指定行がない
Unmatched DATA's values and READ variable	READするDATA文がない
Unmatched IMAGE-spec in USING	USINGのイメージ使用が誤っている
xxx function error	ビルトイン関数のエラー
xxx nest overflow	入れ子 (n e s t) が多すぎる
xxx(xxx) error	PURGE命令の指定ファイルがない
xxx(xxx,xxx) error	RENAME命令の指定ファイルがない
xxx: "xxx" file was opened with xxx mode.	ファイル・ディスクリプタのモード (書き込み/読み込み) が定義したものと合っていない
xxx: CANNOT convert into string	文字列に変換できない
xxx: invalid first type in xxx	コマンドの構文の最初が誤っている
xxx: invalid second type in xxx	コマンドの構文の2番目が誤っている
xxx: invalid source type in xxx	式の代入でソース側のタイプが誤っている
xxx: invalid target type in xxx	代入しようとする変数のタイプが誤っている
xxx: Invalid TERGET operand in XXX	xxxの中の書式に無効なものがある
xxx: Syntax error	文法が間違っている

エラー・メッセージ	内 容
You cannot use POKE command	P O K E 命令を実行しようとした
yy is invalid value in xxx	y y 行目にある x x x の値が無効である
yy: Undefined Control Register	C O N T R O L 命令のレジスタ番号が誤っている
yy: UNIT addr error in xxx	G P I B アドレス指定が誤っている

MEMO 

本製品に含まれるソフトウェアのご使用について

本製品に含まれるソフトウェア（以下本ソフトウェア）のご使用について以下のことにご注意下さい。

ここでいうソフトウェアには、本製品に含まれる又は共に使用されるコンピュータ・プログラム、将来弊社よりお客様に提供されることのある追加、変更、修正プログラムおよびアップデート版のコンピュータ・プログラム、ならびに本製品に関する取扱説明書等の付随資料を含みます。

使用許諾

本ソフトウェアの著作権を含む一切の権利は弊社に帰属いたします。

弊社は、本ソフトウェアを本製品上または本製品とともに使用する限りにおいて、お客様に使用を許諾するものといたします。

禁止事項

お客様は、本ソフトウェアのご使用に際し以下の事項は行わないで下さい。

- 本製品使用目的以外で使用する事
- 許可なく複製、修正、改変を行う事
- リバース・エンジニアリング、逆コンパイル、逆アセンブルなどを行う事

免責

お客様が、本製品を通常の用法以外の用法で使用したことにより本製品に不具合が発生した場合、およびお客様と第三者との間で著作権等に関する紛争が発生した場合、弊社は一切の責任を負いかねますのでご了承下さい。

保証について

製品の保証期間は、お客様と別段の取り決めがある場合または当社が特に指定した場合を除き、製品の納入日(システム機器については検取日)から1年間といたします。保証期間中に、当社の責めに帰する製造上の欠陥により製品が故障した場合、無償で修理いたします。ただし、下記に該当する場合は、保証期間中であっても保証の対象から除外させていただきます。

- 当社が認めていない改造または修理を行った場合
- 支給品等当社指定品以外の部品を使用した場合
- 取扱説明書に記載する使用条件を超えて製品を使用した場合(定められた許容範囲を超える物理的ストレスまたは電流電圧がかかった場合など)
- 通常想定される使用環境以外で製品を使用した場合(腐食性の強いガス、塵埃の多い環境等による電気回路の腐食、部品の劣化が早められた場合など)
- 取扱説明書または各種製品マニュアルの指示事項に従わずに使用された場合
- 不注意または不当な取扱により不具合が生じた場合
- お客様のご指示に起因する場合
- 消耗品や消耗材料に基づく場合
- 火災、天変地異等の不可抗力による場合
- 日本国外に持出された場合
- 製品を使用できなかったことによる損失および逸失利益

当社の製品の保証は、本取扱説明書に記載する内容に限られるものとします。

保守に関するお問い合わせについて

長期間にわたる信頼性の保証、国家標準とのトレーサビリティを実現するためにアドバンテスでは、工場から出荷された製品の保守に対し、カスタム・エンジニアを配置しています。

カスタム・エンジニアは、故障などの不慮の事故は元より、製品の長期間にわたる性能の保証活動にフィールド・エンジニアとしても活動しています。

万一、動作不良などの故障が発生した場合には、当社のMS(計測器)コールセンターにご連絡下さい。

製品修理サービス

- 製品修理期間
製品の修理サービス期間は、製品の納入後10年間とさせていただきます。
- 製品修理活動
当社の製品に故障が発生した場合、当社に送っていただく引取り修理、または当社技術員が現地に出張しての出張修理にて対応いたします。

製品校正サービス

- 校正サービス
ご使用中の製品に対し、品質および信頼性の維持を図ることを目的に行うもので、校正後の製品には校正ラベルを貼付けし、品質を保証いたします。
- 校正サービス活動
校正サービス活動は、株式会社アドバンテス カスタマサポートに送っていただく引取り校正、または当社技術員が現地に出張しての出張校正にて対応いたします。

予防保守のおすすめ

製品にはエレクトロニクス部品およびメカニカル部品の一部に寿命を考慮すべき部品を使用しているため、定期的な交換を必要とします。適正な交換期間を過ぎて使用し発生した障害に対しては、修理および性能の保証ができません場合があります。

アドバンテスでは、このようなトラブルを未然に防ぐため、予防保守が有効な手段と考え、予防保守作業を実施する体制を整えています。

各種の予防保守を定期的実施することで、製品の安定稼働を図り、不意の費用発生を防ぐため、年間保守契約による予防保守の実施をお勧めいたします。

なお、年間保守契約は、製品、使用状況および使用環境により内容が変わりますので、最寄りの弊社営業支店にお問い合わせ下さい。

ADVANTEST

<http://www.advantest.co.jp>

株式会社アドバンテス

本社事務所
〒100-0005 千代田区丸の内1-6-2 新丸の内センタービルディング
TEL: 03-3214-7500 (代)

第4アカウント販売部(東日本)
〒100-0005 千代田区丸の内1-6-2 新丸の内センタービルディング
TEL: 0120-988-971
FAX: 0120-988-973

第4アカウント販売部(西日本)
〒564-0062 吹田市垂水町3-34-1
TEL: 0120-638-557
FAX: 0120-638-568

★計測器に関するお問い合わせ先

(製品の仕様、取扱い、修理・校正等計測器関連全般)

MS(計測器)コールセンタ ☎ TEL 0120-919-570
FAX 0120-057-508
E-mail: icc@acs.advantest.co.jp